2

AD-A256 997

AL-TP-1992-0035-Vol-2

FUNCTIONAL REQUIREMENTS OF AN ADVANCED INSTRUCTIONAL DESIGN ADVISOR: TASK ANALYSIS AND TROUBLESHOOTING (VOLUME 2 OF 3)

ARMSTRONG LABORATORY

Martha C. Polson
Peter G. Polson

Institute of Cognitive Science
University of Colorado
Boulder, CO 80309

David Kieras

College of Engineering
University of Michigan
Ann Arbor, MI 48109

Henry M. Halff

Halff Resources, Incorporated
4918 Thirty-Third Road North
Arlington, VA 22207

Reynold T. Hioki

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
Brooks Air Force Base, TX 78235-5000

412480

92-27289

September 1992

Interim Technical Paper for Period March 1990 - February 1991

AIR FORCE MATERIEL COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5000

NOTICES

This paper has been reviewed and is approved for publication.

[illegible]

REYNOLD T. HIOKI
Project Scientist

[illegible]

HENDRICK W. RUCK, Technical Director
Technical Training Research Division

[illegible]

RODGER D. BALLENTINE, Colonel, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1992	3. REPORT TYPE AND DATES COVERED Interim – March 1990 – February 1991

4. TITLE AND SUBTITLE

Functional Requirements of an Advanced Instructional Design Advisor: Task Analysis and Troubleshooting (Volume 2 of 3)

5. FUNDING NUMBERS

C – F33615-88-C-0003
PE – 62205F
PR – 1121
TA – 10
WU – 43

6. AUTHOR(S)

Martha C. Polson
Peter G. Polson
David Kieras
Henry M. Halff
Reynold T. Hioki

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute of Cognitive Science
University of Colorado
Boulder, CO 80309

College of Engineering
University of Michigan
Ann Arbor, MI 48109

Halff Resources, Incorporated
4918 Thirty-Third Road North
Arlington, VA 22207

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

Armstrong Laboratory
Human Resources Directorate
Technical Training Research Division
Brooks Air Force Base, TX 78235-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL-TP-1992-0035-Vol-2

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Dr. Daniel J. Muraida, (512) 536-2981

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Advanced Instructional Design Advisor (AIDA) is an R&D project being conducted by the Armstrong Laboratory Human Resources Directorate and is aimed at producing automated instructional design guidance for developers of computer-based instructional materials. The process of producing effective computer-based instructional materials is complex and time-consuming. Few experts exist to ensure the effectiveness of the process.

The content of this paper addresses the major implications for instruction based on cognitive and educational research. Principles such as the ones contained in this paper would comprise a substantial part of the knowledge base for the AIDA.

14. SUBJECT TERMS

Cognitive psychology
Cognitive task analysis
Educational psychology
Instructional design
Instructional strategies
ISD

15. NUMBER OF PAGES 124

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF CONTENTS (Continued)

LIST OF TABLES

PREFACE

The work reported herein was done for the Advanced Instructional Design Advisor project at the Air Force Armstrong Laboratory (Human Resources Directorate). The substance of this research was done under contract to Mei Associates, Inc., the primary contractor on the Advanced Instructional Design Advisor (Contract No. F33615-88-C-0003).

This work was done as part of the second phase effort on the Advanced Instructional Design Advisor. The initial phase of this project, documented in AL-TP-1991-0014, established the conceptual framework and functional specifications for the Advanced Instructional Design Advisor, an automated and intelligent collection of tools to assist subject matter experts who have no special training in instructional technology in the design and development of effective computer-based instructional materials. This second phase provided the design specifications for an experimental prototype.

Mei Associates' final report for the second phase is being published as an Armstrong Laboratory Technical Report (AL-TR-1991-0085). In addition, Mei Associates received nine papers from various consultants working on this phase of the project. These nine papers have been grouped into 3 sets and edited by AL/HRTC personnel. They are published as Volumes 1 - 3 of <u>Functional Requirements of an Advanced Instructional Design Advisor</u>:

Volume 1: Epitomizing Functions

Volume 2: Task Analysis and Troubleshooting

Volume 3: Simulation Authoring

This is Volume 2 in the series. Capt Reynold T. Hioki wrote sections I and VI. Drs. Martha C. Polson and Peter C. Polson wrote section II. Drs. Martha C. Polson and David Kieras wrote section III. Dr. Henry M. Halff wrote sections IV and V.

SUMMARY

The Advanced Instructional Design Advisor is an R & D project being conducted by the Air Force Armstrong Laboratory (Human Resources Directorate) in response to an Air Training Command (ATC) Manpower, Personnel, and Training Need calling for improved guidelines for authoring computer-based instruction (CBI) (MPTN 89-14T).

Aggravating the expensive and time-consuming process of CBI development is the lack of Air Force personnel who are well-trained in the areas of instructional technology and educational psychology. More often than not, a subject matter expert with little knowledge of CBI is given the task of designing and developing a computer-based course. Instructional strategies that work in a classroom are often inappropriate in a computer-based setting (e.g., leading questions may work well in a classroom but are difficult to handle in a computer setting). Likewise, the computer offers the capability to present instruction in ways that are not possible in the classroom (e.g., computer simulations models can be used to enhance CBI).

The Advanced Instructional Design Advisor is a project aimed at providing subject matter experts who have no background in computer-based instructional systems with automated and intelligent assistance in the design and development of CBI. The goal is to reduce CBI development time while ensuring that the instructional materials are effective.

I. INTRODUCTION (Hioki)

The Advanced Instructional Design Advisor is an R & D project aimed at providing automated and intelligent assistance to inexperienced instructional designers tasked to design and develop computer-based instruction (CBI). The particular problem addressed by this line of research is the need for more cost efficient methodologies for the design and development of CBI. Current methods for developing CBI are expensive, time-consuming, and often result in ineffective instruction due to the general lack of expertise in computer-based instructional systems (Spector, 1990).

The Advanced Instructional Design Advisor project is divided into four phases:

Phase 1: Conceptualization & Functional Specifications

Phase 2: Conceptual Refinement & System Specifications

Phase 3: Prototype, Field Test, & Refinement

Phase 4: Technology Demonstration & System Validation

The first two phases have been performed under Task Order Contracts. The third phase is being accomplished via a Broad Agency Announcement (BAA). The fourth phase will be performed under a fully specified contract. The work reported herein concerns the second phase.

The next four sections of this paper focus on conceptual approaches to automating the task analysis process and instructional design of maintenance training, specifically procedural and troubleshooting training, which could be implemented in an Advanced Instructional Design Advisor-like system.

In the second section of this paper, Polson and Polson present two theoretical approaches describing the acquisition of proceduralized skills (i.e., Cognitive Complexity Theory (CCT) and ACT*) capable of forming the basis for an AIDA-like system to generate automated procedural training. Both approaches build on the general concepts of production systems. CCT is an extension of the GOMS data representation model while ACT* traces its beginnings to Anderson's general theory of skill acquisition. Other areas addressed include several critical issues concerning instructional effectiveness. Of greatest concern is the need for meaningful and correct task analyses prior to instructional design and development. Without this, instruction is sure to fail. Other issues elaborated on include structuring and presenting domain knowledge, sequencing of instruction, and assessing performance.

A meaningful and correct cognitive task analysis is the cornerstone for the development of any valid instructional course and is equally important for CBI. In section three, Polson & Kieras further elaborate on automating the cognitive task analysis process by discussing the need for capabilities that support the development of mental models and the execution of procedure representations. Specific recommendations include incorporation of a refined GOMS approach with additional capabilities to capture explanation, breadth of component representation and explicit procedures.

In section four, Halff envisions beyond the cognitive task analysis phase and describes the critical instructional design elements needed for the automation of maintenance training. In this approach, Halff emphasizes the importance of appropriate mental models, procedures and troubleshooting skills and their roles as prerequisites in accomplishing maintenance tasks. He closes this section by providing insights to the automation of maintenance training followed by automation of the instructional design process.

Finally, in section five, Halff extends his prior work in section four by instantiating his description of automating maintenance training with the specific instance of operational aircraft maintenance procedures. Within this example, the role of mental models, troubleshooting procedures and problem solving skills become apparent within the basic framework of an AIDA. Halff closes this section by discussing the instructional implications of this approach.

II. AIDA: PROCEDURAL TRAINING (Polson & Polson)

Introduction

This document is intended to summarize the relevant theoretical and empirical knowledge on the acquisition of procedural skills from the cognitive science literature. Of particular interest are those aspects of the literature that would be relevant to constructing a transactional shell that could be used by a person who is a subject matter expert, but a novice instructional designer, to create computer based training materials for teaching procedural skills. We will first cover the background literature, focusing on the recent cognitive science literature, then consider the implications for instructional design.

Foundations of the Cognitive Science Symbolic Information Processing Approach

One aspect of cognitive science that is directly relevant to the task at hand is the symbolic information processing approach that has its foundations in the early problem solving work of Newell and Simon (1972) and the work on human computer interaction by Card, Moran, and Newell (1983). This work formed the basis of later relevant research by a number of investigators, including the ACT* theory of Anderson (Anderson, 1983, 1987; Anderson, Boyle, Corbett, & Lewis, 1990) and the Cognitive Complexity Theory of Kieras and Polson (Kieras & Bovair, 1986; Kieras & Polson, 1985; Bovair, Kieras, & Polson, 1990).

The System Architecture

The clearest summary of the information processing approach as it stood in the early eighties is found in Card et al. (1983). In Chapter 2 of their book, they laid out the basic assumptions about the architecture of the system. Briefly summarized, the underlying assumptions are:

1. The architecture of human cognition is that of an information processor. The components of the processor are:

    a. A PERCEPTUAL system that includes

    - Perceptual memories
    - Perceptual processors

    b. A MOTOR system

c. A COGNITIVE system that includes

- Cognitive memories including

  -- Working memory which contains knowledge currently active.

  -- Long term memory which contains knowledge stored for future use.

- A cognitive processor

2. Notable features of this architecture, which have educational implications, are that 1) working memory is limited in capacity, while long term memory appears not to have capacity limits; 2) the basic unit of the cognitive processor is assumed to be a recognize-act cycle and all cognitive processing is the result of a discrete number of cycles; and 3) on each cycle the contents of working memory activate or recognize associated actions in long term memory which, in turn, changes the contents of working memory.

3. Some basic assumptions about human behavior are that 1) it is rational and goal oriented, i.e. purposeful; 2) all behavior is an instance of problem solving to achieve a goal; and 3) problem solving is a process of moving from some initial state to the desired end state or goal by using various methods, heuristics, or strategies (see also Gagné, 1985; Kintsch, Tennyson, Gagné and Muraida, 1991; Polson, Tennyson and Spector, 1991, and Section III).

Representation of Procedural Knowledge

In addition to summarizing the current view of the architecture of the human information processing system, Card et al. (1983) developed a general representation scheme for the knowledge underlying routine cognitive skills involved in successful use of computers and other complex artifacts. Their representation is called a GOMS model.

The GOMS Model

1. Goals, Operations, Methods, and Selection Rules

The GOMS model (Card et al., 1983) represents a person's knowledge of how to carry out routine skills in terms of *goals*, *operations*, *methods*, and *selection rules*.

Goals represent a person's intention to perform a task, subtask, or single cognitive or physical operation. Goals are organized into structures of interrelated goals that sequence methods and operations. Examples of goals are to locate the fault in a malfunctioning piece of equipment, to align a search processor in a phased-array radar, or to edit a manuscript.

Operations characterize elementary physical actions (e.g., pressing a button, setting a switch, or attaching a probe) and cognitive or mental operations (e.g., perceptual operations, retrieving an item from memory, or reading a voltage and storing it in working memory). The most primitive mental operations are actions such as receiving perceptual information, making a basic decision, depositing facts from working memory into long term memory, retrieving facts from long term memory and activating them in working memory, forming a goal, etc. These primitive operations are the basic operations of the cognitive information processor.

A person's knowledge is organized into methods which can be thought of as subroutines. Methods generate sequences of operations that accomplish specific goals or subgoals. The goal structure of a method characterizes its internal organization and control structure. Examples include methods for diagnosing a fault.

Selection rules specify the appropriate conditions executing a method to effectively accomplish a specific goal in a given context. Selection rules are compiled pieces of problem-solving knowledge. They function by asserting the goal to execute a given method in the appropriate context.

2. The Content and Structure of Procedural Knowledge

The GOMS model assumes that execution of a task or procedure involves decomposition of the task into a series of subtasks. A skilled person executing a procedure has effective methods for each subtask. Accomplishing a task involves executing the series of specialized methods that perform each subtask. There are several kinds of methods. High-level methods decompose the initial task into a sequence of subtasks. Intermediate-level methods describe the sequence of functions necessary to complete a subtask. Low-level methods generate the actual user actions necessary to perform a function.

A person's knowledge of how to do a complex task is a mixture of task-specific information--the high-level methods--and system-specific knowledge, the low-level methods. Intermediate-level methods can appear in many different contexts. For example, to move a section of text using a word processor, a subtask involves selecting the text.

Then, a high-level method would be
- move the text

Intermediate methods would include
- select the text
- cut the text
- locate the place where the text is to be inserted
- insert the text

Low level methods include
For selecting the text
- set the cursor at the beginning of the text to be moved
- press the mouse button
- set the cursor at the end of the text to be moved
- press the shift key and the mouse button simultaneously

For cutting the text
- select cut from the edit menu

In summary, the GOMS model characterizes the user's knowledge as a collection of hierarchically organized methods and associated goal structures that sequence methods and operations. The knowledge captured in the GOMS representation describes both general knowledge of how the task is to be decomposed and specific information on how to execute methods required to complete the task of entering a route.

Although Simon (1975) had earlier used the production system formalism to model problem solving, Card et al. (1983) only noted that the cognitive processor could be modeled as a production system without elaborating the point. Since then, using the GOMS model as a foundation, production system models have been developed for various cognitive processes including the use of text editors (Kieras & Polson 1985, Singley & Anderson, 1989), problem solving (Karat, 1983;), and text comprehension (Kieras, 1982). Anderson (1982, 1983) extended Newell and Simon's (1972) approach and developed a general theory of cognition, called the ACT* theory, based on a production system formalism. Kieras and Polson (1985) proposed in a framework called the cognitive complexity theory, CCT, that the knowledge represented in a GOMS model be formalized as a production system. Before elaborating this work further, we will give a brief overview of a production system formalism.

Production System Models

A production system is made up of three components:

1. a collection of *rules* describing the knowledge required to perform a task;

2. a *working memory* containing a representation of a user's immediate goals, intermediate results generated during the

execution of a method, and a representation of the external behavior of the system; and

3. an *interpreter* that controls execution of the rules.

A rule is a condition-action pair of the form

IF (condition) THEN (action)

where the condition and action are both complex. The *condition* represents a pattern of information in working memory that specifies when a physical action or cognitive operation represented in the *action* should be executed. The condition includes a description of an explicit pattern of goals and subgoals, the state of the environment, (e.g., the state of the switches and other information on a piece of equipment), and other needed information in working memory (e.g.,the desired voltage reading or some other parameter).

ACT* Theory

Anderson (1983, 1987) has the broadest theory of cognition in which there has been a serious attempt to relate the principles of the theory to the acquisition of proceduralized skills. This application of his ACT* theory is very nicely elucidated in three recent papers (Anderson, 1987, 1990; Anderson et al., 1990).

An integral part of Anderson's theory of skill acquisition is the distinction between declarative and procedural knowledge, a distinction which was first made in the field of Artificial Intelligence over a decade ago (e.g., Winograd, 1975) and has important educational implications. In the ACT* theory knowledge of facts, rules, concepts, past events, etc. are all aspects of declarative knowledge. This knowledge is characterized by the fact that we have conscious access to it, it can be encoded relatively quickly, and it can be acquired without any particular commitment as to how it will be used. Declarative knowledge may be acquired in numerous different ways; reading text, watching film, a conversational discourse, everyday experiences, etc. (Anderson et al., 1990).

The acquisition of declarative knowledge must precede the acquisition of procedural knowledge. In the ACT* theory procedural knowledge is acquired through the interpretative application of declarative knowledge. This process involves trial and error practice. Once acquired, procedural knowledge is embodied in a highly use specific way (Anderson et al., 1990).

Fundamental to the distinction between declarative and procedural knowledge is the assumed difference in the underlying representation. In the ACT* theory and other theories dealing primarily with factual knowledge (van Dijk and Kintsch, 1983, Kintsch, 1988), declarative knowledge is represented as a network of labeled nodes. The labeled connections each have some level of strength or activation associated with them which denotes the strength of the association, either negative or positive. The nodes may be single propositions, concepts, instances of a concept, plans, or schema.

Procedural skills and other cognitive skills are generally represented as productions (however, see Schneider & Detwiler, 1988, for a different view). Productions are the key element of Anderson's view of skill acquisition. Cognitive skills of which procedural skills are an example are modeled by a set of productions which are hierarchically arranged. Productions form both the cognitive units and specify the hierarchical goal structure that organizes the problem solving. Working memory contains the active knowledge that is available at the time. When a new goal, such as "solve this algebra problem" is established by some external event, then this goal becomes active in working memory. If this goal is the condition part of an established "macro" production for solving that type of problem, which exists in production memory, then that production will be applied and a solution generated to the problem. However, if a known solution does not exist, then a search must begin for a solution. The assumption is made that people solve novel problems by applying weak-problem solving methods to the declarative memory that they have about this domain. These weak-problem solving solutions include analogy, means-ends analysis, working backward, hill climbing, and forward search (Newell & Simon, 1972).

Using these weak problem solving rules, which are not specific to any domain, a solution to the problem is generated. Which weak method will be used is determined by what declarative knowledge exists about the domain. As the problem is solved by successively searching through declarative memory for the necessary conditions, a trace of the hierarchically organized productions is produced. The learning mechanism is a compilation process that creates efficient domain specific productions from this trace.

Proceduralization is the first aspect of this compilation process. As the production trace is generated in the production memory, it is now no longer necessary to search declarative memory for the conditions; therefore they no longer have to be held in working memory, thereby reducing the processing load and decreasing the likelihood of an error.

The next stage is compilation. The series of productions are compiled into a single production which accomplishes the same task. Compilation speeds up performance considerably. This compiled production is now available to produce solutions to future problems of this nature. This highly compiled knowledge is not open to introspection. The intermediate steps are no longer accessible without some type of reconstruction process.

Both declarative knowledge and productions are strengthened with use or practice. The strongest relevant declarative knowledge is most likely to be in working memory. Stronger productions are more rapidly matched to the contents of working memory. Strength primarily affects speed, but has a secondary effect on accuracy (Anderson et al., 1990).

Anderson has developed intelligent tutoring systems for LISP and Geometry that are based on this account of learning (Anderson, Boyle, Farrell, & Reiser, 1984). These tutors have been very successful. In fact, the LISP tutor is one of the few tutors which evaluation has shown to be successful. From his work on tutoring, Anderson has derived several principles for instruction that are derived from the ACT* theory (Anderson, 1987; Anderson et al., 1984; Anderson et. al., 1990). These will be discussed in the implications section later.

Cognitive Complexity Theory

Another line of research which has implications for the training of procedural skills is the cognitive complexity theory (CCT) of Kieras and Polson. Kieras and Polson (1985) assume that *cognitive* complexity of a task determines the difficulties in acquisition, transfer, and retention of the skills necessary to perform the task. Their work is done in the domain of using a computer system, but the results can apply to other systems, particularly computer based equipment, which is becoming increasingly common. Cognitive complexity is a function of the *content*, *structure*, and *amount* of knowledge required to perform a task using a specific system.

The theoretical foundations for the CCT work on the analyses of acquisition, transfer, and retention is the GOMS model (Card et al., 1983). Both CCT and the GOMS model characterize the knowledge necessary to perform routine cognitive skills, such as operation of a text editor. The GOMS formalism describes the content and structure of the knowledge underlying these skills. CCT represents this knowledge as production rules permitting quantification of the amount of knowledge. CCT incorporates all of the assumptions of the GOMS model. Quantitative predictions of training time, transfer of skills, and performance can be derived from the production rule formalism (Polson, 1987; Polson & Kieras, 1985; Polson, Muncher, & Engelbeck, 1986).

Procedural Learning and Transfer of Training

The dominant theoretical approach for explaining specific transfer effects is based on a framework proposed by Thorndike and Woodward (1901) and Thorndike (1914). In this approach, transfer between two tasks is mediated by the number of common elements assumed to be stimulus-response associations. Associations acquired in a first task that successfully generalize to a second do not have to be relearned during the acquisition of the second task. Only those associations unique to the second task have to be acquired. If a large number of the associations required to successfully perform the second task transfer from the first task, there can be a dramatic reduction in training time compared to the training time required for the second task without experience on the first.

Both Kieras and Polson (Kieras & Bovair, 1986; Polson & Kieras, 1985) as well as Singley and Anderson (1989) have proposed that a common elements theory of transfer could account for positive transfer effects during the acquisition of operating procedures. They assumed that the common elements are individual rules. For instance, in a computer system with a consistent interface such as the Macintosh, common methods are used to achieve the same goals even when these goals occur in different tasks. These shared methods are represented by rules common to models of the different tasks. It is assumed that these shared rules, once learned, are always incorporated into the representation of a new task at little or no cost in training time. After a user has had some experience with an application program with a consistent user interface, learning a new task requires the acquisition of a small number of unique rules. These new rules may be a small fraction of the total number of rules necessary to execute the new task. Rules representing the common methods transfer and do not have to be relearned. In many areas, these common methods can be a large part of the knowledge required to perform the new task.

The transfer processes outlined above are the basis for derivations of models fit to data derived from transfer experiments (Bovair et al., 1990; Polson et al., 1985; Singley and Anderson, 1989). These transfer processes incorporate three strong assumptions. First, there is no characterization of nonspecific transfer effects, (e.g. generalized practice effects); it is assumed that improvements and performance are mediated by common rules. Second, these shared rules are recognized and utilized in novel contexts. Third, common rules can be incorporated into the representation of a new task at no cost in training time. This theory has implications for how procedural knowledge should be represented, the most efficient order for the acquisition of skills, as well as other aspects of procedural training which will be discussed in the next section.

Implications for a Transaction Shell on Procedural Learning

Task Analysis and Representation of Knowledge

The major difficulty that will be faced in developing a successful transactional shell for developing courseware for procedural learning skills will be performing a successful task analysis. It has been widely recognized in the educational literature since Gagné (1968) that successful instruction in a complex task involves decomposing the task into a collection of *meaningful subtasks*. Modern cognitive theories (Bovair et al., 1990; Card et al., 1983; and Newell & Simon, 1972) all require that a theorist provide a *correct* goal structure for a complex task in order to be able to successfully model the acquisition and performance of that task.

The terms *meaningful* and *correct* are what make things difficult. A complex task can obviously be divided up into an arbitrary collection of hierarchically arranged components. A meaningful and correct decomposition reflects the actual underlying structure of the task in a comprehensible manner that can be learned by a student. For example, a given subtask cannot be arbitrarily complex. If it contains more than four or five major components the subtask in turn must be decomposed into a collection of sub-subtasks (Kieras, 1988a).

Experts have a large amount of knowledge about relevant task decompositions in the domain of their expertise. However, much of this knowledge is implicit, i.e. highly compiled, which presents a problem for instruction. The task of decomposing the knowledge and making it explicit so it could be incorporated successfully in instructional manipulations is a very different skill from the skill of successfully performing the task.

The problem of doing a successful task analysis in collaboration with the subject matter expert who is not a trained educator or cognitive scientist is equivalent to the problem of developing an expert system where the domain expert is not a knowledge engineer, but must communicate his or her knowledge to the knowledge engineer in order for it to be incorporated into the expert system. This process is long and tedious and is often a severe test of even successful collaborations. During an initial phase the domain expert describes to the knowledge engineer his/her understanding of the knowledge underlying his/her performance. The knowledge engineer then attempts to incorporate this description into an expert system. The initial version of this system is incomplete; it contains errors because the domain expert may not have explicit understanding of the expertise, and as a result of miscommunications between the domain expert and the knowledge engineer. Successive iterations of the developing expert system require that the domain expert gain a more explicit understanding of his expertise and that the knowledge engineer require extensive knowledge of the application domain. It should be clear why this is a long and painful process.

We have the identical problem in performing the critical useful task analysis that is necessary to develop successful instructions in the procedural skills necessary to carry complex maintenance tasks and other skills of interest to the Air Force, with the added difficulty that we want a machine to be able to extract the domain knowledge. Modern cognitive theory provides us with a partial solution to this very difficult problem and suggests the design of knowledge acquisition tools that can be incorporated into the transactional shell.

The goal structure of the GOMS model is the modern representation of such a successful task analysis. Since the GOMS model has been rigorously formalized as a production (Bovair et al., 1990) and Kieras (1988a) has developed a set of explicit heuristics for doing a GOMS analysis, we are in a position to incorporate these results into a transactional shell that would support doing task analysis by deriving an appropriate goal structure for the skill being tutored. Kieras' (1988a) heuristics could be incorporated into the shell and coach a domain expert during the process of deriving the goal structure from a task analysis. Thus, if the domain expert defined a single subtask that violated the complexity criteria for a subtask, i.e. more than five steps, the shell could prompt the instructional designer with a suggestion, 'consider decomposing the current subtask into a collection of less complex sub-subtasks'. Furthermore, the tool could also check that the developing task decomposition was syntactically correct.

Decomposition of Procedural Knowledge for Training

The task decomposition discussed in the preceding section is the major critical component of decomposing the knowledge necessary to perform a complicated task into meaningful subtasks. The next issue to be followed up is the grain size of the actual steps in the procedure. Here again, we can look to the cognitive complexity theory (Bovair et al., 1990) for very explicit rules. Furthermore, these rules are supported by an extensive body of empirical evidence (Anderson et al., 1990; Bovair et al., 1990; Polson & Richardson, 1988; Singley & Anderson, 1989). Anderson (1983) argues that an individual production is the cognitive unit. Thus, the learner ultimately has to be provided with the information necessary to derive the sequence of productions necessary to carry out a task. Anderson also advocates that the student model be represented as a production system in an intelligent tutoring system. We are not advocating this here. However, a rich collection of theoretical results provides us with very precise answers as to grain size of the instructional material, which may need to be adjusted as the student progresses (Anderson et al., 1984).

The task analysis, a subgoal decomposition using the GOMS terminology, decomposes the top level goal of performing the task into a collection of subtasks and their associated goals. The decomposition terminates in the sequence of physical actions and elementary cognitive operations necessary to accomplish each of the lowest level subtasks. Kieras (1990) provides some very specific heuristics for making decisions about this grain size. The general result seems to be that physical actions that take less than a second and cognitive operations that take from a tenth of a second to a second represent the appropriate grain size. Bovair et al. (1990) argues that in a novice a single rule represents a single physical action or cognitive operation.

Obtaining the single production grain size may not be as difficult as it sounds. Following is a section of the Technical Order 31P1-2FPS85-52 on adjusting the search processor power supplies in a phased array radar.

5-41. SEARCH PROCESSOR ALIGNMENT

5-42. GENERAL. The search processor alignment consists of power supply adjustments, power supply monitor adjustments, and precise adjustment of search processor channels and associated logic and monopulse circuits.

NOTE

Accurate alignment of the search processor requires that the search digital processor be aligned prior to aligning the search analog processor. Refer to T. O. 31P1-2FPS-85-282 (Paragraph 5-17, SEARCH DIGITAL PROCESSOR ALIGNMENT).

5-43. TEST EQUIPMENT REQUIRED.
The following test equipment, or suitable equivalent, is required to perform the search processor alignment:

NOTE: *30 item equipment list omitted as not relevant to point*

5-44. POWER SUPPLY ADJUSTMENTS.
Perform the following procedure to adjust the search processor power supplies (PS1 through PS5). Ensure that the search processor is in an off-line cleared status prior to making any adjustments.

NOTE

If CURRENT LIMIT potentiometer adjustments do not result in a drop off (step e below) set this control to approximately 1/8 turn from the fully counterclockwise position.

1. PS1 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS1.
   b. On power supply PS1, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust -12 VDC PS1 VOLTAGE ADJUST potentiometer for 12 vdc as indicated on front panel meter.
   e. On PS1, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

2. PS2 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS2.
   b. On power supply PS2, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust -12 VDC PS2 VOLTAGE ADJUST potentiometer for 12 vdc as indicated on front panel meter.
   e. On PS2, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

3. PS3 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS3.
   b. On power supply PS3, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust -6 VDC PS3 VOLTAGE ADJUST potentiometer for 6 vdc as indicated on front panel meter.
   e. On PS3, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

4. PS4 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS4.
   b. On power supply PS4, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust +6 VDC PS4 VOLTAGE ADJUST potentiometer for 6 vdc as indicated on front panel meter.
   e. On PS4, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

5. PS5 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS5.
   b. On power supply PS5, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust +12 VDC PS5 VOLTAGE ADJUST potentiometer for 12 vdc as indicated on front panel meter.
   e. On PS5, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

5-45. POWER SUPPLY MONITOR ADJUSTMENTS. Perform the following procedure to adjust the search processor power supply monitors (PS1 through PS5) and voltage regulator monitor (VR1).

NOTE

Complete paragraph 5-44, power supply adjustments, prior to performing this procedure. Additionally, ensure that all power supply IN TOLERANCE (green) lamps are illuminated prior to performing each power supply monitor adjustment.

These technical orders are, for the most part, at the level of the single production. However there are only minimal hints as to the goal structure. In this case, the primary task of the analysis module of the transaction shell would be to extract the goal structure from the subject matter expert and arrange the steps within that hierarchial structure. For a novice, even the individual switch adjustments which contain six steps are too long for a single goal (see Bovair et. al., 1990; Kieras, 1988a).

Presenting Declarative Knowledge in a Way that Facilitates the Learning of Procedures

There are two problems in providing explicit instruction on a complex procedural skill. The first, and most obvious, is simply the sheer volume of knowledge to be learned. A complex skill will probably be represented by several hundred rules. For relatively simple tasks, up to about 50 rules, the training time per rule seems to range between 20 to 40 seconds when using a very intrusive constraining tutorial procedure that optimizes rote learning. (We are not advocating this training process.) The figure of from 20 to 40 seconds ignores the extra training that would be required to eliminate confusion between highly similar rules and to learn rules to a criterion that would permit them to be retained over days or months or years. (Bovair et al.; 1990; Olson & Olson, 1990; Polson et al., 1988).

The other problem concerns the concepts that are incorporated into the goal and subgoal descriptions. These concepts are part of the clarity of knowledge that an expert has about the task domain. They also may refer to an expert's extensive knowledge of how a device to be maintained or troubleshooted actually works. Thus, a top level subgoal in some complex maintenance procedures might be to align the RF oscillator. These are all meaningful concepts to the expert. To the novice they are probably just a collection of nonsense syllables. The difficult question then becomes how do we provide the necessary instruction in this declarative knowledge, the concepts that make meaningful the elements of the goal structure. Anderson et al. (1990) and Kieras, (Kieras, 1990; Kieras & Bovair, 1984, 1986) have advocated developing text for teaching the declarative knowledge from a complete simulation of the task, thereby incorporating all the necessary knowledge for the task, but eliminating any extraneous information. To the extent that the task analysis module produces a reasonably explicit analysis at the appropriate grain level, this may be a feasible approach. However, it should be noted that researchers who have concentrated on the acquisition of declarative knowledge place more emphasis on teaching the student the necessary monitoring and strategic skills to enable them to examine their own knowledge structures for completeness and providing support to make the nature of the knowledge structures evident than presenting a pre-digested text (Campione & Brown, 1990; Kintsch et al., 1991; Nathan, Kintsch, & Lewis, 1988; Palincsar & Brown, 1984; Scardemalia et al., 1989). This is an area that could use further research.

Training Should be in the Context of Actually Performing the Procedure of a Simulation of that Procedure

This statement is somewhat weaker than the recommendation from the intelligent tutoring community (Anderson et al., 1984; Frederiksen & White, 1990) which states that learning should be problem centered or that instruction should be in the problem solving context. We use the weaker version because the stronger version may indeed require an intelligent tutoring system. However the higher level goal should be strived for whenever possible. In the technical order example given above, it would not take much machine intelligence to give instruction on the first switch setting and let the trainee attempt to set the next switch without further instruction on a computer based simulation. From the transfer of training approach there is a minimum amount to be learned in that transfer. For computer based training of procedural skills training in the context of actually performing the task means that at least some minimal level of simulation of the task will need to be available. The typical subject matter expert/novice instructional designer in most Air Force fields is not likely to have extensive programming skills, therefore some type of easy to use simulation package will need to be accessible from the transaction shell. This leads back to one of the first author's favorite hobby horses, the need to provide guidance on graphics.

Performance Assessment

Declarative knowledge of a skill is a necessary, but not sufficient, condition for actually acquiring the skill. Therefore, assessment techniques such as multiple choice questions, which assess only declarative knowledge, are not adequate for testing if a procedural skill is learned. The actual performance of the task or a simulation thereof will need to be assessed (for a discussion see Frederiksen, 1990; Frederiksen & White, 1990). Correct performance of a task is also not a sufficient condition for determining that training is adequate. Note that in the Anderson framework a procedure can be correctly performed before compilation occurs. Additional practice will be needed after initial successful performance to progress from proceduralization to compilation. After compilation still further practice will be needed to strengthen the activation level of the procedure, which both makes it faster to execute and decreases the amount of working memory resources that it requires (Anderson et al., 1984; Anderson et al., 1990). At this point speed of correct performance will be a better decision criteria for determining when training should be terminated than accuracy (Anderson, 1990; Schneider, 1985).

Minimizing Working Memory Load

All of the approaches discussed stress the necessity to minimize the load on working memory when training procedural skills. Recommendations for achieving this goal include:

1. Provide some means, either graphic or otherwise, of making the trainees aware of where they are in the goal hierarchy. Goals serve as memory aids to help the trainees remember where they are and what is to be done next. Relieving the trainee of the necessity of keeping this information in working memory frees up resources to devote to other aspects of the learning task. (Anderson et al., 1984; Scardemalia et al., 1989).

2. Ensure that earlier steps are sufficiently mastered (i.e. practiced) so that they will require few working memory resources. (This recommendation is related to the automaticity issue which will be briefly discussed later.) This leaves working memory capacity to incorporate new productions for compilation.

3. Order training such that a minimum number of new productions are to be acquired at one time (see also the next section).

Training Sequence for Procedures for Efficient Learning

In the Kieras and Polson CCT model learning time for a subtask is determined by the number of new productions to be learned. Therefore to minimize frustration levels and keep motivation levels high, whenever possible, new tasks to be learned should be arranged so that minimum amount of new information needs to be acquired at each step. The cognitive science approaches also stress the necessity for providing support for successful performance before the entire skill has been acquired (Anderson et al., 1984; Anderson et. al., 1990; Frederiksen & White, 1990). Anderson et al. (1984) state this principle as "enable the student to approach the target skill by successive approximation" (p 5).

Feedback

Timing of feedback on errors

One controversial conclusion that Anderson and his colleagues concluded from their work on tutors is that feedback on errors needs to be as immediate as is possible, i.e. as soon as the error is diagnosed (Anderson et al., 1984; Anderson, 1990; Reiser, Kimberg, Lovett, & Ranney, 1989). This view is based on the fact that the student is building a trace of the necessary productions to solve the problem as he works. Errors and digressions only slow the process of building the correct trace. Also when students get lost, they use tremendous amounts of cognitive resources to get back to their original goals (Anderson, 1982).

However, other studies of computer-based feedback cast some doubt on how immediate the feedback should be. Schmalhofer, Kuehn, Messamer, and Charron (1989) have shown that a "selective" tutor--one that provides feedback only after two consecutive errors have occurred--had certain advantages in introductory LISP learning and problem solving over an "immediate" tutor--the type advocated by Anderson and his colleagues.

Similarly, Lee (1989) showed that delayed feedback in a tutor for solving genetics problems can lead to better overall problem solving than immediate feedback, especially on novel or difficult problems. Immediate feedback in these domains may lead to "blind learning" where the student can use trial and error to complete a problem, but has no notion of the reasoning or principles underlying a solution. Since the reasoning may not be overt in the error correction, it is difficult for the student to learn it. Also students may need to be allowed time to self-correct, since self-generated answers are better remembered than those supplied externally. (See Anderson, 1983, chapter 4 for a discussion of this issue.) Timing of feedback may also need to be adjusted as a function of the experience level of the student. Novices may like and need immediate feedback, while more experienced students find it annoying (Anderson et al., 1990).

Other aspects of feedback

A number of researchers in the cognitive sciences area have addressed the area of the type of feedback that should be given. Although the conclusions are stated in a number of different ways, basically the feedback should be designed to encourage the student to think through the processes necessary to arrive at the correct answer, rather than just supplying the correct answer (Anderson et al., 1984; Frederiksen & White, 1990; Scardemalia, Bereiter, McLean, Swallow, & Woodruff, 1989).

There is sufficient controversy concerning the timing and nature of feedback that more research is needed in this area before a definitive recommendation can be made.

Automaticity and Procedural Training

Another aspect of cognitive science which is relevant to the training of procedural skills is the literature on automaticity. (See for example Regian & Schneider, 1990 and Schneider, 1985). The issues to which this literature is relevant include:

1. How to assess when training should be terminated.

2. What types of tasks can be automated and will therefore benefit from extended practice?

3. The order in which tasks that require the performance of more than one skill at once should be trained.

The simulation techniques in this field are an example of the nonsymbolic, connectionist tradition in cognitive science and will require additional background material that could not be addressed in the time allotted in this cycle.

III. TASK ANALYSIS IN AIDA (Polson & Kieras)

Introduction

The first step in the design of any instruction is a task analysis to determine what should be taught. From the cognitive science information processing approach, it is argued that a behavioral analysis is not sufficient. A cognitive analysis needs to be performed because education and training should take into account the cognitive processes involved in learning and performance, not just the objective behaviors required (See Glazer & Bassock; 1989, as well as several chapters in Psotka, Massey, & Mutter, 1988 for recent discussions of this issue).

Section IV of this report identifies three types of cognitive structures important to the maintenance enterprise: the execution of procedures, a mental model of the equipment, and fault isolation skills. Thus an adequate cognitive task analysis should identify the information and skills that must be imparted to the student to support the acquisition of these cognitive structures. Due to time limitations in this paper, I will only address the areas of procedures and mental models.

The current walk-through of generating, with an Advanced Instructional Design Advisor System (AIDA), training materials on the T-38A aircraft is partly based on two cognitive task analyses performed by David Kieras of the University of Michigan. The first, presented in Kieras (1988b), centered on the issue of what mental model should be taught concerning the engine ignition system of the T-38A aircraft and was used in constructing the example training materials in the RAPIDS II Authoring Manual. The second, generated by a subcontract to this effort, is a cognitive task analysis of the troubleshooting procedures for the fault of "no start" in the T-38A engine (See Appendix A).

While the current walk-through is based in part on Kieras' cognitive task analysis, little attention has been paid to date to specifying the nature of the task analysis in the AIDA system. The primary focus of this paper is to explore the task analysis conducted by Kieras as a basis for specifying the task analysis requirements in an AIDA designed for maintenance training. The questions to be considered include

- What is the nature of the cognitive task analysis?
- How detailed does the analysis need to be?
- How should that task analysis be represented in AIDA?
- How do you map the representation onto the instructional materials?
- What kinds of aids and/or guidance could be provided to a novice instructional designer, who might also be a subject matter expert (SME), but a novice instructional designer, to perform the task analysis?

This paper does not attempt to provide complete or final answers to the above questions, but primarily strives to spell out the issues that need to be addressed and some of the relevant literature.

Cognitive Analysis of Procedures

The analysis of the troubleshooting procedures done by Kieras is a particular type of analysis known as a GOMS (Goals, Operators, Methods, and Selection Rules) analysis, which derives from the Cognitive Complexity Theory (CCT) of Kieras and Polson, (Bovair, Kieras, & Polson, 1990) and has as its intellectual predecessor the work of Card, Moran and Newell, (Card et al., 1983)[1]. This approach entails analyzing the tasks to be accomplished into a meaningful series of goals and subgoals. Each goal to be accomplished is recursively broken into a series of subgoals until a level is reached in which accomplishing the subgoal can be achieved by either a primitive level motor or mental act. Such a simple act for the T-38A start system would be *press the left start button*, or *apply the shorting stick*.

Goals represent a person's intention to perform a task, subtask, or single cognitive or physical operation. Goals are organized into structures of interrelated goals that sequence methods and operations. An example goal from troubleshooting the engine would be to determine if the ignition system is functioning correctly.

Operations characterize elementary physical actions (e.g., pressing a button, setting a switch, or attaching a probe) and cognitive or mental operations (e.g., perceptual operations, retrieving an item from memory, or reading a voltage and storing it in working memory). The most primitive mental operations are actions such as receiving perceptual information, making a basic decision, depositing facts from working memory into long term memory, retrieving facts from long term memory and activating them in working memory, forming a goal, etc.

Methods generate sequences of operations that accomplish specific goals or subgoals. The goal structure of a method characterizes its internal organization and control structure. The GOMS model assumes that execution of a task or procedure involves decomposition of the task into a series of subtasks. A skilled person executing a procedure has effective methods for each subtask. A novice may have less efficient methods. Accomplishing a task involves executing the series of specialized methods that perform each subtask. There are several kinds of

[1]This approach was discussed in the previous section, but I will cover the basic description again, so that this section can be read apart from the previous one.

methods. High-level methods decompose the initial task into a sequence of subtasks. Intermediate-level methods describe the sequence of functions necessary to complete a subtask. Low-level methods generate the actual user actions necessary to perform a function.

A person's knowledge of how to do a complex task is a mixture of task-specific information--the high-level methods--and system-specific knowledge, the low-level methods.

Then, a high-level method for troubleshooting the T-38A engine would be

- check out the starting operations of the engine

Intermediate methods which are part of the high level method for checking out the symptom of no start would include

- check for bad ignition
- check for fuel flow problem
- check for defective starting system
- check for altitude limitation problem

Low level methods for the intermediate methods include

For checking for bad ignition
- apply shorting stick to AB plug
- check the ENGINE IGNITION, R AUTOSYN INST & IGNITION INVERTER circuit breakers for proper engagement

For fuel flow problem
- check fuel system circuit breakers for proper engagement

Selection rules determine which method to select. In an expert, selection rules are compiled pieces of problem-solving knowledge. The selection rule must state the appropriate context for using any given method. If there is more than one method, the rule must state when each method is appropriate.

In summary, the GOMS model characterizes the user's knowledge as a collection of hierarchically organized methods and associated goal structures that sequence methods and operations. The knowledge captured in the GOMS representation describes both general knowledge of how the task is to be decomposed and specific information on how to execute the methods required to complete the task.

One of the greatest advantages of this approach for our purposes is that Kieras has prepared a detailed guide for doing task analysis of procedures using the GOMS methodology (Kieras, 1988a)[2]. He has also defined a language call (NGOMSL) or "Natural" GOMS Language which is relatively easy to read and write. Kieras' guide also includes procedures for doing a GOMS analysis by using a breadth-first expansion of methods rather than trying to describe goal structures directly.

Mental Models Analysis

Section IV of this report summarizes the importance for maintenance training of imparting correct and adequate mental models of the equipment. Kieras (1988b, 1990) pointed out that the most accurate way of determining the mental model to be taught would be to do a complete cognitive simulation. However, realizing that this is not always a feasible approach, Kieras (1988b) spelled out some heuristics that could be used to determine the mental model that should be taught in lieu of a complete simulation. The heuristics are:

- relevance to task goals
- accessibility to use
- critical procedures and inference strategies

All of these heuristics involve doing an analysis equivalent to a GOMS analysis of the task at hand. In addition, two other hierarchical cognitive analyses are required: an explanation hierarchy and a hierarchical decomposition of the device structure and mechanisms. The *relevance to task goals* heuristic states that explanations should only be given if they are relevant to a task goal. To carry out this heuristic an explanation hierarchy is constructed. The first pass at what goes into this hierarchy can be what is in the existing documentation. The goals of the GOMS analysis are then mapped to the explanation hierarchy, which will reveal any missing explanatory information as well as any extraneous material which need not be taught. Constructing the explanation hierarchy is not really extraneous work since this material is needed for the instructional material. For instance, this is the material that goes into the message windows in the Rapids system.

The second heuristic, *accessibility to use*, implies that the device illustration or simulation which is presented to the technician should not contain parts which he cannot access. Again, this involves a mapping of the GOMS analysis, but onto the device description, rather than the explanation hierarchy.

[2]An unpublished 1990 edition of this document can be obtained from David Kieras, Technical Communications Program, TIDAL Bldg. 2360 Bonisteel Blvd, University of Michigan, Ann Arbor, MI 48109-2108.

The third heuristic says that the GOMS analysis should be examined for procedures that will be difficult to learn due to what appears to be arbitrary content. These procedures should then be analyzed to determine what inferences would need to be made in order for the content to appear logical rather than arbitrary. The information necessary to make those inferences should then be made explicit in the training materials. This information will need to be included either in the explanation hierarchy or the device description.

Level of Detail of the Task Analysis

Kieras (Kieras, 1990) as well as Anderson (Anderson, Boyle, Corbett & Lewis, 1990) have advocated doing a complete cognitive simulation of a given task which is based on a cognitive analysis of the task in order to determine the content of instructional materials and training procedures. The advantage of a simulation is that it ensures that the analysis is complete down to the level of simple operations or operators for most aspects of the task. The information that can be derived from the simulation includes the time to learn the task, the amount of transfer of training from one procedure to another, and the execution time for various procedures or methods. The disadvantage is that a complete cognitive simulation requires a tremendous amount of effort to implement, even after the cognitive analysis of the content of the instruction is complete. However, as can be seen from the GOMS analysis of Kieras in Appendix A, the use of the GOMS method for cognitive analysis of procedures does not require that it be followed through by a complete simulation or that all tasks be analyzed to the level of simple operators.

How low level the analysis needs to be for the procedures for any given instructional package will be determined in large part by the level of expertise of the trainees. For instance, for the problem of No Start with the Probable Cause of no ignition or poor ignition[3],the first step is to check igniter plugs for firing and proper spark rate. This is followed by a note that the proper spark rate is 3 sparks in 2 seconds (See Figure 2). Presumably this is as low as the analysis needs to go. In terms of a computer based instructional system, this detail could be represented by clicking on a designated igniter plug icon handle (handles are mouse sensitive areas) which will give its status. If the status had been set to bad then the simulation would continue with the procedure, (the next step if the igniter plugs do not fire is to check the static inverter). The actual motor and perceptual operations necessary in checking the spark rates would not have to be explicitly laid out. However, for a novice technician some of the steps in the T-38A troubleshooting manual

[3](Page 6-7 of Technical Manual for Engine Conditioning of the T38)

such as *remove the engine* do seem rather high level and may need to be broken down into subtasks. Anderson (Anderson, Boyle, Farrell, & Reiser, 1984) refers to this as adjusting the grain level of the instruction.

As a way of decreasing the workload of authoring the simulation and/or doing the GOMS analysis for a given domain, a library of generic low level procedures such as testing igniter plugs (as well as their corresponding simulations) could be provided in an AIDA configured for that domain. In fact these could be a set of separate modules that are given as screening tests to insure that these low level methods or methods which occur in many different troubleshooting situations, such as *remove the engine* are learned before entering simulations which are higher level or aimed at specific problems. A problem for a generic system, as opposed to a system written explicitly for a domain such as electronic maintenance or airplane maintenance, is knowing what skills and knowledge can be assumed. If the domain is known, there is probably a reasonably finite set of testing skills, mechanical procedures, etc. that are known to be required to perform the task. For instance, if the student is said to be at such and such a skill level in a particular field, is there a list of basic procedures that the student can be expected to know and which would not have to be represented in detail in a particular domain?

Representing The Task Analysis

In the CCT approach of Kieras and Polson, a simple production system is used to implement the results of a NGOMSL analysis into a working simulation. The device knowledge necessary to carry out the simulation is represented in a Generalized Transition Network (GTN)[4](Kieras & Polson, 1985). However, a number of representation schemes are possible. A scheme used by Anderson in his PUPS system is a candidate representation that is probably compatible with the Transaction Shell representation discussed by Merrill (Jones, Li, & Merrill, 1990b). Anderson's PUPS (Penultimate Production Systems) theory holds that procedures are acquired by compiling declarative knowledge (Anderson et al., 1990). The declarative knowledge necessary for compiling the procedures which model the task performance is represented in schema based structures called PUPS structures. These schema include slots for the function of the entity being represented by the schema, a form slot for the physical appearance of the entity, and a precondition slot which states the preconditions necessary for the function to be achieved (Anderson et al., 1990). In compiling the productions which are the basis of procedural knowledge, the function slot maps to the goal to be

[4]An example of the representation of a device in the GTN formalism is given in Kieras, 1990.

achieved which will require knowledge of the entity represented; the preconditions slot maps onto the condition of the condition-action pair in a production; the form slot in the PUPS tutors holds the form of the current action to be carried out such as a particular LISP function. A similar scheme could be used for representing the GOMS analysis. Merrill has proposed an activity frame that has paths or sequences of actions. This frame could also have slots for the function, the operators, and the outcome. The values for these slots could probably be automatically generated from a NGOMSL analysis just as it is technically feasible to generate a running production rule based simulation from a NGOMSL analysis.

The explanation hierarchy can be represented in numerous different ways. From my limited understanding it appears that the representation scheme already proposed by Merrill for AIDA (Hickey, Spector, & Muraida, in press) would be adequate to represent the explanation hierarchy. The device knowledge will ultimately be represented in the graphical simulation. The initial representation may be a hierarchical listing of the names of the device components or perhaps a block diagram, which can serve as a guide for constructing the sketch which will guide the construction of the graphical simulation.

Mapping the Content of GOMS and Mental Model Analysis to the Device Simulation

Following Kieras' approach will yield three hierarchically arranged representations. The GOMS analysis will spell out the steps to be followed in carrying out procedures for operating, calibrating, troubleshooting, or repairing the equipment starting with the highest level goals and methods. These are successively decomposed to lower level subgoals and methods. The GOMS analysis will also identify any device components that need to be included in the representation of the device structure as well as the declarative knowledge that needs to be conveyed about them; function, location, name, etc. The explanation hierarchy will contain the causal and declarative knowledge necessary to execute the procedures, support inferences necessary for constructing a mental model of the equipment, and define the attributes and rules of objects, etc.

The device simulation in a system such as Rapids contains a graphic representation of the device structure and qualitative simulations of its functioning. Authoring in the Rapids II simulation starts with a temporary sketch which is derived from the prior cognitive analysis, particularly the mental model analysis which entails interrelating the GOMS analysis, the explanation hierarchy and the hierarchical device structure decomposition. However the construction of the simulation is done in bottom up fashion starting with the lowest level of the device hierarchy. The lowest level objects are the bottom items in the device structure analysis. These correspond to the objects manipulated by the lowest level operators in the GOMS model. For this reason, it is not feasible to develop the simulation and do the GOMS analysis and explanation hierarchy in parallel, which might be tempting to the novice instructional designer, who wants to get on with "real" work. The analyses have to be complete before the construction of the simulation can begin.

The behavior of the objects are defined by attribute handles and rules. These aspects of the simulation are drawn from the explanation hierarchy. Once the basic simulation is complete, procedures which are carried out on the device are authored by carrying out a sequence of actions which correspond to actions spelled out to accomplish the goals in the GOMS analysis. The individual actions correspond to the operators. What is missing from the simulation representation is any indication of the function or purpose, i.e. goals, of the procedure. These have to be represented in the dialogue windows.

Aids for Doing a Cognitive Analysis

As mentioned in the previous section, it should not be difficult to implement a shell which can guide a novice in doing a GOMS analysis of a particular task using either the documentation at hand or the knowledge of a subject matter expert. The shell can be based on the previous work of Kieras (Kieras, 1988a) who has invested a large amount of time in writing a manual on how to do GOMS analysis and in developing an English-like language for representing the analysis. Included in the guide are many rules of thumb which could be implemented in a knowledge based shell to give guidance to the SME or instructional designer. For instance, Kieras recommends that a given method contain no more than five steps. If there is more than that some may need to be grouped into a higher level method. There is also guidance on creating generic methods to represent methods which occur often in slightly different context. For instance, rather than a method for checking each specific circuit breaker, there would be a check circuit breaker method, which has as a variable which circuit breaker to check. This variable information is held in working memory.

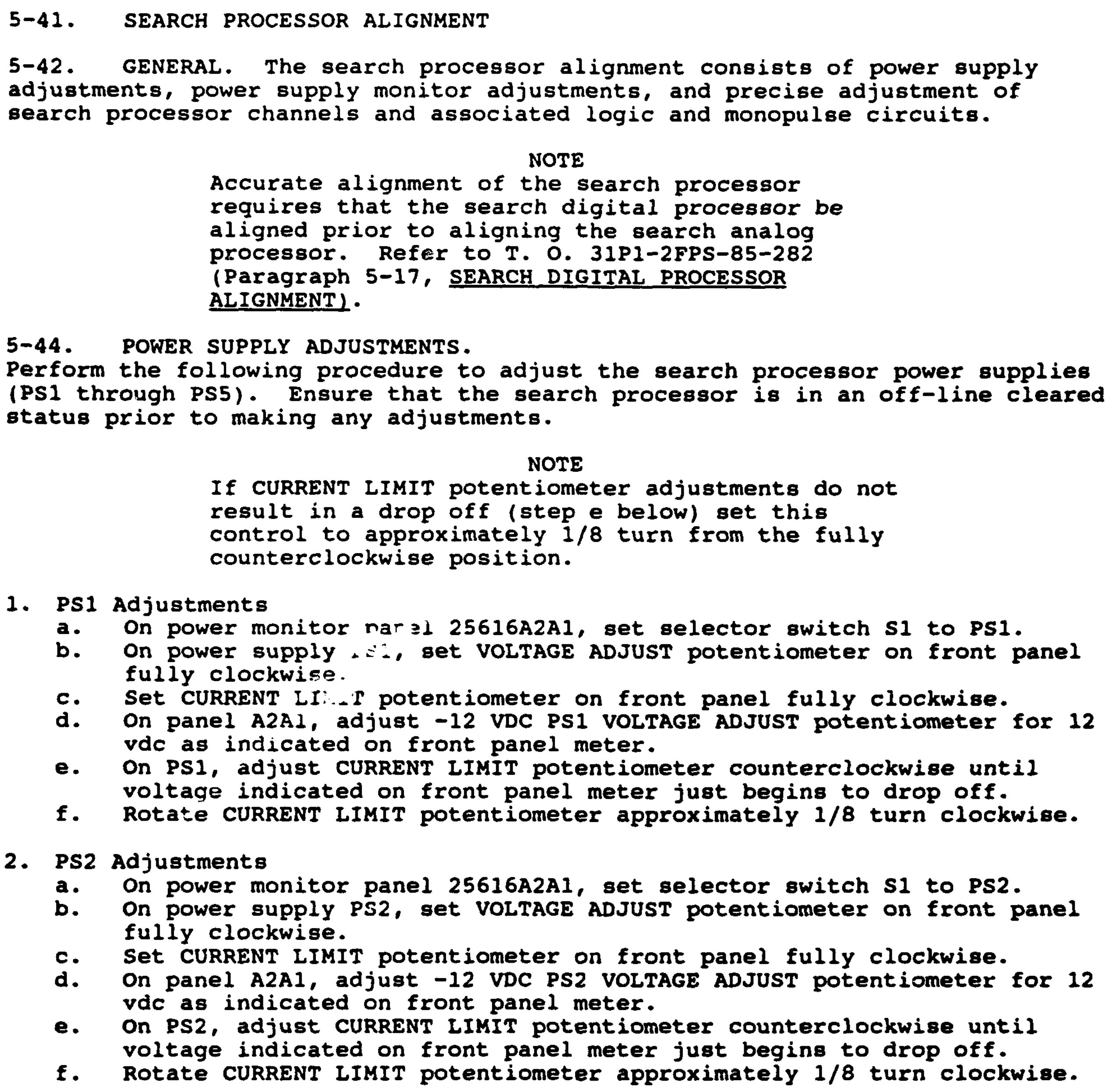
This shell could do much of the bookkeeping necessary for a GOMS analysis such as creating a list of methods and information identified by the methods that need either already to be known or taught, such as their location, etc. A more sophisticated shell could automatically map the results of the analysis into the knowledge representation system. A less sophisticated system would create a paper guide for what should be hand entered into the representation system. Similar shells could also be created for the explanation hierarchy and the device structure and function knowledge. However, I am not aware of any explicit guidelines for doing such analyses.

How difficult a given task analysis will be and the type of guidance that will be needed will depend to a large part on the nature of the documentation. The diversity of the document is illustrated by the two excerpts from Technical Orders given in Figures 1 and 2. Figure 1 shows a section of Technical Order 31P1-2FPS85-52 on adjusting the search processor power supplies in a phased array radar. The steps in carrying out a procedure are spelled out in excruciating detail. However, it lacks any hint of a goal structure or any supporting material for creating a mental model which could guide the performance of the task. The problem in doing a cognitive analysis of this task would be providing this information in the form of the higher level goals and methods and the explanation hierarchy. It could be noted that this particular document could have benefitted from the heuristic of presenting a general method and noting with variables to which segments of the equipment it would apply with variables.

Figure 2 is an excerpt from Technical Order 1T-38A-2-6-2 for engine conditioning of the T-38A aircraft. It goes to the other extreme in that single steps are the level of *remove the engine*. However, the general goal structure is represented in the trouble and probable cause headers which are always visible.

Conclusions

I recommend that the task analysis approach developed by Kieras and his colleagues be adopted for the task analysis module of AIDA. This includes a GOMS analysis for the procedural aspects of the task and performing a mental model analysis by way of developing an explanation hierarchy and a decomposition of device structure and function and relating them to the GOMS analysis. Developing shells to aid in the cognitive task analysis is technically feasible. However, a great deal of care will need to be taken to be sure that the shells are implemented in such a way that the instructional designer perceives them as an aid, not a hindrance or an extraneous useless requirement. The task analysis can be represented in such a way as to be compatible with the current conceptualization of the AIDA representation and the Rapids II formulation. However it may not be feasible to have a single representation which serves all aspects of the AIDA system unless a totally new system is designed which incorporates the ideas of the current components but not the current implementations.



5-41. SEARCH PROCESSOR ALIGNMENT

5-42. GENERAL. The search processor alignment consists of power supply adjustments, power supply monitor adjustments, and precise adjustment of search processor channels and associated logic and monopulse circuits.

NOTE

Accurate alignment of the search processor requires that the search digital processor be aligned prior to aligning the search analog processor. Refer to T. O. 31P1-2FPS-85-282 (Paragraph 5-17, <u>SEARCH DIGITAL PROCESSOR ALIGNMENT</u>).

5-44. POWER SUPPLY ADJUSTMENTS.
Perform the following procedure to adjust the search processor power supplies (PS1 through PS5). Ensure that the search processor is in an off-line cleared status prior to making any adjustments.

NOTE

If CURRENT LIMIT potentiometer adjustments do not result in a drop off (step e below) set this control to approximately 1/8 turn from the fully counterclockwise position.

1. PS1 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS1.
   b. On power supply PS1, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust -12 VDC PS1 VOLTAGE ADJUST potentiometer for 12 vdc as indicated on front panel meter.
   e. On PS1, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

2. PS2 Adjustments
   a. On power monitor panel 25616A2A1, set selector switch S1 to PS2.
   b. On power supply PS2, set VOLTAGE ADJUST potentiometer on front panel fully clockwise.
   c. Set CURRENT LIMIT potentiometer on front panel fully clockwise.
   d. On panel A2A1, adjust -12 VDC PS2 VOLTAGE ADJUST potentiometer for 12 vdc as indicated on front panel meter.
   e. On PS2, adjust CURRENT LIMIT potentiometer counterclockwise until voltage indicated on front panel meter just begins to drop off.
   f. Rotate CURRENT LIMIT potentiometer approximately 1/8 turn clockwise.

Figure 1. A Section of Technical Order 31P1-2FPS85-52 on Adjusting the Search Processor Power Supplies in a Phased Array Radar.

Trouble	Probable Cause	Isolation and Remedy
6-5. No start	1. No ignition or poor ignition	1. Check igniter plugs for firing and proper spark rate. Spark rate should be 3 sparks in 2 seconds.
		NOTE AB igniter plug can be checked by looking up the tailpipe. Main igniter plug can be checked by shorting out AB plug with a piece of brass or steel wool attached to a dry wooden stick and listening for main igniter plug to fire.
		2. If igniter plugs do not fire, check static inverter electrical power by actuating fuel/oxygen test switch and checking fuel quantity indicator operation.
		3. Check ENGINE IGNITION & R AUTOSYN INST and IGNITION INVERTER circuit breakers for proper engagement.
		NOTE If steps 2 and 3 indicate that static inverter is not operating properly, check inverter system in accordance with T. O. 1T-38A-2-7.
		4. Apply external electrical power to aircraft, and check igniter plugs for firing.
		5. If plugs do not fire with external power applied to aircraft, check for 115-volts ac electrical power at pin N in engine and accessories disconnect plug.
		6. If electrical power is not available in pin N, check aircraft electrical system in accordance with T. O. 1T-38A-2-7.
		7. If electrical power is available at pin N, problem is in the engine. Remove engine from aircraft.

Figure 2. An Excerpt from Technical Order 1T-38A-2-6-2 for Engine Conditioning of the T38A Aircraft.

IV. AUTOMATING MAINTENANCE TRAINING (Halff)

Introduction

This section is concerned with instructional design for maintenance training. My aim is to identify some common elements in the design of maintenance training, to suggest how maintenance training can be automated with interactive media, and to discuss the potential for automated development of interactive maintenance training materials.

Maintenance training is not one, but is rather a large class of training enterprises, of which the instances are individual training courses. These courses vary from equipment to equipment. Within the same class of equipment, courses vary according to the type of maintenance required—bench versus depot, for example. Even within these constraints courses can vary according to instructional objectives. Courses for novices address different objectives than courses for experienced maintainers. Courses that address a class of equipment will have different objectives than courses solely concerned with one model.

Nonetheless, there are enough commonalities among maintenance training courses to make maintenance training something of a natural concept. An effective characterization of the class of maintenance training courses should aid in the design (by humans or computers) of particular courses.

In the first part of this section, I offer such a characterization in three steps. I first describe the tasks that constitute effective maintenance. I then describe the mental structures that support proficiency in these tasks. Finally, I take up the training regimes that support development of the mental structures required for proficiency. The treatment proposed is more descriptive than analytic. Thus, I make no claim for the uniqueness of any part of this characterization. To some extent, it reflects current research, but it is also based on traditions developed within the maintenance training community itself.

The second part of the section is concerned with automation of the training process and the development of materials. My aim in this second part is to suggest a design for automated, interactive maintenance training based directly on the results of the first part, and to determine which aspects of this design can be automatically implemented by computers.

The section concludes with a summary and some general observations on maintenance training as an example training domain.

Instructional Design for Maintenance Training

Maintenance Tasks

I begin with a rough list of the tasks that a maintainer must master in order to effectively maintain a piece of equipment. The tasks described here are equipment oriented in that they denote what must be done to the equipment. They do not address issues of how the tasks are accomplished by the maintainer.

Operation

In most maintenance contexts the maintainer must be able to operate, to some degree, the equipment being maintained. Operational skills are used to verify the status of the equipment, to prepare the equipment for maintenance, and to interpret reports from operators.

Calibration and Adjustment

Many devices must be configured for particular operating environments, calibrated, and adjusted on occasion. Maintenance personnel are routinely called on to effect such adjustments. These adjustments are often a part of preventive maintenance, and they often constitute repairs.

Testing

Equipment testing is a critical part of maintenance. Maintainers must be able to test an equipment's operational status. They must also be able to conduct particular diagnostic tests during the course of troubleshooting. These tests often require the use of general-purpose and specialized test equipment, and this test equipment must itself be properly calibrated and operated.

Access and Disassembly

In the course of repair, testing, and calibration, maintainers must gain access to particular components for observation and manipulation. The procedures used to gain access can be straightforward in some cases. In others, special procedures are required to ensure that gaining access to one part of the equipment will not damage other parts. These procedures are normally specified by the manufacturer of the device.

Repair

By repair, I mean the operations needed to return a device to operability once a fault has been isolated. Repairs therefore include replacement of faulted components, cleaning, adjustment, patching, and a host of other operations.

Troubleshooting

Perhaps the most challenging maintenance operation, from a training viewpoint, is that of troubleshooting. Troubleshooting is the process of identifying the physical cause (*fault*) of an existing or potential *malfunction* of the equipment's operational capabilities. For the most part, troubleshooting takes place after a malfunction occurs, but troubleshooting also comes into play when a test—say, during preventive maintenance—reveals a potential fault.

Cognitive Components of Maintenance Skills

We pass now to the mental structures that support the maintenance tasks described above. There are any number of schemes for representing cognition (Anderson (1983) for example, or Jones, Li, and Merrill (1990a, 1990b)) each seeking to distinguish itself from the others by using different terms for the same concepts. Since my aim in this part is to provide a useful cognitive framework for discussing maintenance training, I make no apology for borrowing freely from the cognitive representation schemes known to me or for ignoring those not familiar to me.

Three different types of cognitive structures seem to me to be important to the maintenance enterprise. A *mental model* of the equipment figures heavily in virtually all maintenance operations. Most of the tasks described above require the execution of fixed *procedures*. A third component of effective maintenance are *fault isolation* skills.

Mental Models of Equipment

Mental models, in the context of maintenance, refer to the cognitive structures used to reason about the equipment being maintained. In thinking about mental models of a device it is important to take account of the device's structure, its function and its physical manifestation.

1. Structure. An oversimplified view of structural knowledge can be cast in terms of the equipment's *components* and its *topology*. On this view, the device is represented as a directed graph with individual components at the nodes. Each component is represented by a *device model* used to derive the components outputs on the basis of its inputs (or, more precisely, as a function of inputs and changes therein). Components can operate in any of a number of modes, including fault modes, so that differential predictions can be derived for faulted and not faulted cases.

Reasoning within this model takes the form of propagating changes from component to component. Informed of an input or change in input to one component, the model derives the consequences for that component's outputs and then propagates the result to the components connected to the original component's outputs. The general term for this interpretation process is *qualitative reasoning*, as opposed to *quantitative* reasoning which derives predictions from the joint application of mathematical constraints (e.g., Kirchoff's law).

This model, although it forms the basis for some interesting work in maintenance training (Johnson, 1988), has some fundamental inadequacies.

One of the most problematic difficulties with the simple conception is discussed extensively in de Kleer and Brown (1983, 1985). De Kleer and Brown point out that for certain common devices, in particular those involving feedback loops, the simple interpretation process described above arrives at an impasse that can only be resolved by either (a) referring to the device's function, or (b) bringing certain logical constraints to bear on the problem. Since (a) is clearly inappropriate in the case of malfunctioning devices, some consideration must be given to the joint functioning of logic and qualitative reasoning in determining how mental models support troubleshooting.

A second problem with the qualitative reasoning approach, and one not nearly so well explored in the literature, is that of bridging the gap between competence and performance models. The mental models of real technicians, even skilled ones, are always incomplete, and their reasoning processes are governed by the same working memory constraints that govern all human mental activities. Thus, mental models, as they function in real life, are almost certainly structured to permit the technician to deal with the device in workable chunks. Maintainers learn to isolate faults, for example, in larger modules, and then to deal with components within those modules. The key to understanding how devices are actually represented in technician's heads probably lies in the study of how technicians comprehend technical documentation and the physical layout of the devices.

2. Function. Functional knowledge of a device (and its subsystems and components) is even less well understood. From an extreme qualitative reasoning viewpoint, functional knowledge is irrelevant, for if the structure of a device is well understood, its functional characteristics can easily be derived. Intuition argues, however, that functional knowledge forms the cornerstone of structural knowledge. Functional knowledge can explain or rationalize the structure of a device and thus serve, at least, as a mnemonic for device structure. (For the importance of mnemonics in cognitive skills, see Chase and Ericsson (1982).) In addition, as was mentioned above, functional knowledge may be used to resolve impasses in qualitative reasoning. At the least, functional knowledge guides the technician's evaluation of the

device's operational status. Research on mental models of device functionality is sparse indeed. One of the most interesting analyses of this sort can be found in Kieras (1988).

3. Imagery. Mental models are of little use unless they can be correlated in some fashion with the actual equipment. There is considerable evidence (Kosslyn, 1980) that imagery plays a significant role in our making sense of the outside world. Indeed, some assumption about imagery underlies any depiction of the actual equipment—pictures, high fidelity simulators, the equipment itself—in training and documentation. Imagery also plays an important role in providing internal cognitive support for the mental model. Graphics are pervasive in maintenance training and documentation, and for good reason. For they provide concrete imaginal representations of the abstract notions that make up the working part of the mental model.

4. The Importance of Mental Models. We conclude this part with a few thoughts on the role of mental models in maintenance. Kieras (1988) and Kieras and Bovair (1984) provide an interesting analysis of the importance of mental models in device operation, and many of these arguments apply also to maintenance. Looking back to the list of tasks identified above, we can see the pervasive support provided by mental models.

a. As Kieras and Bovair point out, mental models provide constraints that enable effective *operation* of the equipment with less than perfect memory of the operating procedures themselves.

b. *Calibration and adjustment* are, by definition, goal-oriented procedures that are supported by a knowledge of the structural relations between controls and indicators.

c. While some aspects of *testing* relate directly to the equipment's functionality, most tests are conducted to establish the condition of particular components or subsystems. In these cases mental models provide coherence to the formulation and interpretation of tests.

d. The artifactual nature of equipment means that its conceptual structure is at least roughly reflected in its physical structure. Single circuit boards, for example, often implement single modules. Thus, a mental model of the device offers considerable advantage to the technician who must *disassemble* that device to *gain access* to a particular subsystem, module or component.

e. Whenever a technician faces some choice in making a *repair*, her mental device model can be used to guide that decision. To take a simple case, a basic knowledge of how conductors function makes it easier to determine how to repair an open circuit.

f. *Troubleshooting* often amounts to testing hypotheses about the location of a fault. Mental models support this hypothesis-testing procedure by providing predictions from various hypotheses.

One might argue that any one of these tasks or functions could be carried out without reference to a mental model. However, the large and pervasive advantage that mental models confer on overcoming inherent cognitive limitations render absurd the notion that they should not form the basis of maintenance training.

Procedures

Examining the list of tasks previously identified, one cannot help but be struck by the extensive procedural requirements of the maintenance enterprise. One is easily left with the impression that most of maintenance is nothing more than the execution of simple procedures. (We can take, as a rough measure of a procedure's complexity, the extent of branching or number of choice points in the procedure.)

1. Procedure Definition. Perhaps the first consideration in dealing with this wide range of procedures is the question of definition. Fortunately, the cognitive-science community has devoted considerable effort to this question. My view of an appropriate definition for procedures has three parts (which happily correspond to the three aspects of mental models).

a. A procedure's *control structure* defines the sequence of steps to be taken when executing the procedure. A number of devices are available for representing control structure, including augmented transition networks, and-or graphs, production systems, and others. Whatever the mechanism, it should have the capability to represent both branching (decisions) and sequential constraints (stepwise progression).

b. A procedure's *function* defines its teleology or goal. This term, like that of device function (See "Mental Models of Equipment" above) is theoretically dispensable (and, indeed, may never become apparent to some learners), however, most would argue that an understanding of a procedure's function is an indispensable part of its definition. At the least, functional knowledge supports the use of the procedure in problem solving activities.

c. Procedures also interact with the outside world. In human terms, therefore, the *perceptual-motor* (input-output) concomitants of a procedure are an important part of its definition. When a procedure calls for, say, setting switches or reading dials, the technician must know how to accomplish these actions and observations.

It is also important to recognize that procedures take their inputs from and have their effects on knowledge structures, and in the case of maintenance, on the technician's mental model of the device. Thus any definition of a procedure that does not precisely identify the knowledge structures interrogated or affected is less than complete. The same can be said of procedure definitions that refer to undefined knowledge structures outside of any proposed mental model of the equipment.

2. Procedure Implementation. Clearly, effective execution of any procedure depends on getting procedural knowledge into the head of the technician, but how this knowledge gets in, when it gets in, and in what form are open questions dependent on the nature of the maintenance enterprise.

In some cases procedures must be fully represented in procedural form in the head of the maintainer. To take an extreme case, as soon as the fire light goes on in the cockpit of an aircraft, the pilot becomes a maintainer, and his initial procedure for dealing with the situation must be highly automated.

In other cases, procedures should be *interpreted*, usually from descriptions found in *job aids* and other documentation. Both preventive maintenance and repairs in steam power plants are heavily guided by these sorts of procedures. The skills involved in interpreting these procedures are different, but no less demanding than the skills represented by fully internalized procedural knowledge.

In still other cases, procedures are needed that are not available through training or documentation but rather are invented by technicians to accomplish particular aims. For example, combat has been known to inflict widespread damage to weapon systems and their subsystems The task of dealing with such widespread damage is not covered in detail in either training or documentation (if only because of the astronomical numbers of possible situations). Nonetheless competent technicians can often deal with these situations by redesigning the system on the basis of a mental model.

The invention of procedures to meet certain goals is a special case of problem solving, and the situations that require this sort of problem-solving in the maintenance arena are of two types: complex malfunctions and difficult repairs.

Complex malfunctions include such cases as multiple faults, intermittent malfunctions, faults in systems to complex for standard troubleshooting practices (e.g., feedback systems), unreliable test equipment, and parts inventories contaminated with faulty components. These problems have many of the characteristics of decision making under uncertainty, but I suspect that few decision-theoretic strategies are brought to bear on them in actual maintenance situations.

Difficult repairs can occur when large portions of a system are damaged, when parts needed for particular repairs are not available, when the equipment or parts thereof are not accessible, or when other circumstances prevent normal repairs. Solutions to these problems often involve jury rigs in which some part of the system is redesigned and rebuilt to restore partial or complete functionality.

There are a number of considerations that bear on the choice of procedure implementation. For example, if fast execution of a procedure is required then learning is the preferred route. If a set of procedures is essentially open, then one must rely on the technician's ability to compose them. Unfortunately, knowing that a procedure is a maintenance procedure or even knowing which of the tasks described above it supports does not do much to inform these implementation decisions.

Fault Isolation

Troubleshooting skills are of major concern in most maintenance-training communities. It is therefore not surprising that these skills have received considerable attention in the psychological literature. What is surprising is that a fairly coherent picture of skilled troubleshooting has emerged from these studies. I have discussed this picture in some detail earlier (Halff and Spector, 1991), and will only summarize here.

Troubleshooting, according to this picture, is a problem in a space based on the mental model described above in "Mental Models of Equipment," where devices are viewed as networks of components. This model is configured with one or more components in fault modes, and the troubleshooting problem is to devise a sequence of actions that isolate and repair the faulted components. The actions in the solution sequence can be defined according to the following possibilities.

1. Observe the outputs of some components.

2. Observe the states of some components (e.g., LEDs).

3. Manipulate the states of some components (e.g., switches).

4. Replace certain components.

Costs, in terms of time and money, can be assessed for each of these actions.

The psychology of troubleshooting, conforms (fortunately) to the general picture of skilled problem solving that has arisen in recent years. In particular, troubleshooting skills are viewed as a mix of strong, knowledge-based methods and weak, context-independent skills.

Knowledge based methods are manifested as associations between familiar patterns of observations and troubleshooting actions. To take an example from Hunt and Rouse (1984),

> IF the engine will not start and the starter motor is turning and the battery is strong,
>
> THEN check the gas gauge.

Context free methods are manifested as analysis of the topology of the device to determine which troubleshooting actions discriminate among a set of plausible or hypothetical faults. Again, taking an example from Rouse and Hunt,

> IF the output of *X* is bad and *X* depends on *Y* and *Z* and, *Y* is known to be working,
>
> THEN check *Z*.

In addition to these two selection principles, troubleshooters are also guided by the information-theoretic value of potential observations, and will choose those that provide the greatest reduction in uncertainty (Towne, Johnson, & Corwin, 1983).

Formal *competence* models of the troubleshooting process are available (Hunt and Rouse, 1984; Towne, Johnson, and Corwin, 1983) and are eminently suitable for use in training (Towne and Munro, 1988; Towne, Munro, Pizzini, Coller, and Wogulis, 1990). What is not available at this time are valid *performance* models of troubleshooting. For example, it is obvious that skilled troubleshooters do not compute the information associated with each potential observation in order to choose among them. Rather, they probably use heuristics based on a structured view of the device topology. Sorely needed is a theory of how technicians arrive at this structured view, and how they interpret the view in the course of choosing observations.

To summarize this discussion, we have identified three important aspects of the cognition of maintenance skills. A **mental model** the equipment serves to support virtually all maintenance tasks. This model represents the topology of the device and the behavior of individual components, and it supports causal or qualitative reasoning about the device's behavior. **Procedures** are pervasive elements of most maintenance tasks. However, the content of these procedures and their implementation vary so much from one maintenance domain to another that little can be said about the nature of maintenance procedures per se. A third component of the psychology of maintenance is the process of fault isolation or **troubleshooting**. Models of this process conform to general notions of skilled problem solving as a combination of weak and strong methods. Precise, formal models of the troubleshooting process are available, but they fail to capture some of the details of implementation constraints imposed by human information-processing limits.

Instruction

We turn now to the instruction appropriate for promoting the acquisition of the maintenance skills described above. Instruction is dictated not only by the instructional objectives outlined above but also by the nature of the learning process and by ad hoc circumstances governing the learning context. Nonetheless, it is useful to think in terms of a curriculum that corresponds directly to the structure of the instructional objectives.

Orientation to the Equipment

In this part we consider what instruction is appropriate to convey a mental model of the equipment being maintained. Above, we delineated the main components of such a model: the device's function, including its operation, and the device's structure, including its topology and component behavior.

1. Device Function and Operation. Device function can be conveyed to students in any number of ways.

The device can be identified as one of a larger class of devices whose functions are known to the student. What distinguishes the device from others of its class should also be made known to the student.

The device, or a model thereof, can be shown in operation, preferably under student control, and preferably in such a way as to cover all of the major operating states of the machine.

2. Device Structure: Topology and Component Behavior. As with device function, any aspect of a device's structure can be identified as member of a larger class with the same structural

characteristics. Of more interest are mechanisms for directly conveying device structure. Traditional methods rely on paper documentation and consist, usually, of a block diagram, a narrative description, and data on the specifications of each component. More recently, computer simulations such as STEAMER (Hollan, Hutchins, and Weitzman, 1984), have offered a number of benefits not to be found in conventional documentation. Among these are:

a. support for practice of causal reasoning,

   Students can be given partial information about the state of the device and asked to predict some aspect of device state not evident in the display. Both the sequence of exercises of this sort and the structure of feedback can be based on the structure of the device itself.

b. exhibition of complex component functions,

   Students can be shown (interactively) how a component or subsystem reacts to different combinations of inputs and changes in inputs. For example, students can be given some of the inputs to a component and asked how the others might be set to put the component in a particular state.

c. exhibition of component, subsystem, and device behavior under any normal or faulted operating states,

   Simulations of the sort described can be designed to reflect the conceptual structure of the device. Thus, for example, components that are structurally related can be shown in the same display even though they are physically separate in the equipment itself.

d. association of the mental model's components with their imaginal manifestations,

   Video and other devices can be exhibited in connection with their conceptual counterparts in a simulation. For example, students can be given short drills on the sounds or oscilloscope patterns characteristic of particular states of a device. Other exercises (e.g., Li & Merrill, 1990) can be used to teach the physical locations of a device's components.

e. presentation of mnemonics and other mechanisms for cognitive support of learning.

   Symbolic depictions of device components can be designed to reflect their state or function. The traditional use of icons in electronic diagrams is an obvious example of this mechanism.

Procedures

In the subsection above entitled "Cognitive Components of Maintenance Skills - Procedures," we delineated three different ways of implementing procedures:

- directly from the technician's procedural knowledge,
- from interpretation of a job aid, or
- through composition as the result of problem-solving activity.

We also noted that, however a procedure is implemented, attention must be paid to the *control structure* of the procedure, its *function* and its *perceptual-motor* aspects. Because our characterization of maintenance procedures cannot be much more specific than this, little can be said about instructional methods. The following training suggestions are nothing more than general recommendations for procedure training.

1. Procedural Knowledge. Certain well-known prescriptions apply to the teaching of procedures (in the sense of helping the student achieve unaided execution of the procedure).

a. Depict the control structure of the procedure along with its function, and the constraints that the function imposes on the control structure.

b. Make intermediate results available to students during training.

c. Provide practice in the procedure under conditions that preserve a consistent mapping of stimulus to response.

d. Provide examples that span the space of choices that must be made in executing the procedure.

e. Provide enough practice with real equipment or high-fidelity simulators to ensure mastery of the sensory-motor aspects of the target procedures.

If a procedure has more than one choice point, make sure the student is able to make each choice in isolation of the others.

2. Interpretation Skills. Training technicians to interpret written instructions is (or should be) more a matter of designing the instructions than training the students. The task of interpreting instructions is partly one of inducing the procedure

from examples and partly one of deducing the procedure from a written description.

a. Devise a procedure for interpreting the set of job aids used on the job and teach this procedure using the principles suggested above for Procedural Knowledge. The procedure should function to find the right job aid for the occasion and properly interpret the aid.

3. Problem Solving Skills. Problem solving, of the sort discussed above in "Procedures" is almost always taught on the job, where most of the opportunities for training of this sort occur. Acquisition of problem solving skills can be promoted in school in several ways.

a. Since effective problem solving depends crucially on a technician's facility with the mental model of the system being maintained, extensive practice with this model should enhance her ability to create effective procedures in unanticipated situations.

b. Case study, at least by default, is the preferred method for promoting problem-solving expertise. In many situations, it should not be difficult to import this method from the job site to the school room by recording and packaging selected cases.

c. Certain general problem-solving techniques (e.g., decision theory, heuristic reasoning) seem to describe effective solutions so well that they bear considerable promise in instruction. Configuring these methods to difficult maintenance problems might yield substantial benefit.

Troubleshooting

Troubleshooting is generally taught through a series of troubleshooting exercises, and practice will no doubt be the backbone of the most effective methods of troubleshooting training. This said, what needs to be specified are the sequence of problems to be used in troubleshooting practice and the practice environment.

1. Problem Selection in Troubleshooting Practice. Consistent with the discussion in the above subsection entitled "Fault Isolation," the aim of troubleshooting practice is to build both knowledge-based and context-independent skills. This goal suggests that certain problems, which exercise neither type of skill, should be excluded from consideration in practice sets, and that other problems should be included. In Halff et al., (1991), I suggest a set of simplifying restrictions that exclude non-instructive problems. To quote from that paper,

In typical training situations, certain simplifying assumptions govern the behavior of the equipment.

- Every malfunction is the result of a single faulted component, although in real equipment multiple faults often occur.
- Faults can be characterized as a change in the state or possible states of a component, not in the topology of the equipment, although in real equipment faults can change the nature of the connections among components.
- Neither testing nor replacing a component will fault another component, although in real equipment a faulted component can protect another component from damage.
- Finally, we assume that there are no faulty replacements, even though real world technicians will on occasion return a faulted component to inventory.

In addition to these exclusion criteria, the instructional objectives delineated in "Fault Isolation" suggest that certain types of faults should always be included in troubleshooting practice.

a. Mission critical malfunctions should be included so that students learn the pattern of observations associated with these faults.

b. Principles of discrimination learning dictate that close relatives of mission critical malfunctions should be included in practice sets. These close relatives are those whose patterns closely match those of the corresponding mission-critical faults.

c. To exercise context-free troubleshooting skills, students should be given a set of structure spanning malfunctions. By this, I mean that the faults should be chosen that allow for application of all context-independent strategies.

2. The Practice Environment. The practice environment is as critical to successful training as is the selection of problems. Many general principles of procedure learning also apply to troubleshooting practice.

a. Density of practice is important. Simulators should be used and designed to engage students in as many practice problems as possible.

b. Hidden cognitive operations should be made evident to the student. Thus, the student should be forced to track, actively or passively, the results of each troubleshooting action. Typically this means he should identify which faults are eliminated by each action.

c. Strategic information should be made evident to the student. This can be accomplished though the advice and critiques of a human or machine tutor.

Curricular Issues

The suggestions made above for meeting general instructional objectives present a significant challenge to curriculum design. The objectives themselves are interrelated and so it seems that the curriculum should reflect this interrelatedness. Any of the instructional paradigms that address the objectives can be configured in many different ways. Different components can be chosen for exercises, different degrees of support can be provided. The division of effort between student(s) and instructor can vary. It may therefore be of some use to recall a few general principles that have guided curriculum development.

1. Lesson Structure. The course should be divided into discernable lessons and/or other recognizable units. The goals of each lesson and the methods used to achieve those goals should be made clear to students.

2. Teach Prerequisites First. The subskills or prerequisites of a skill should be addressed in training before the skill itself. Thus, for example, since troubleshooting involves certain inferences to be made from a mental model of the device, exercises addressing these inferences should be provided prior to troubleshooting training.

3. Whole- and Part- Task Training. There are two approaches to training tasks that involve distinct subtasks. *Part-task* training calls for practice of the subtasks in isolation. The *whole-task* approach allows students to practice the entire task using external cognitive support for subtasks that have not yet been mastered. Maintenance training can, and should employ both methods. For example, whole task training in diagnostic maintenance can focus on replacement procedures by allowing an expert to walk the student through the troubleshooting stage of a repair. Part-task training in troubleshooting itself may be of some benefit because eliminating non-essential tasks such as equipment disassembly can increase the density of practice.

4. Completeness. When procedures or problem-solving skills involve branching, exercises should be provided that address all significant variation in input to these procedures. Thus, students should be given troubleshooting exercises that introduce them to all significant topological patterns in the structure of the equipment.

5. Fading. Any exercise can be configured with a variety of cognitive supports. Among these are techniques that exhibit the correct moves to the student, those that provide structural aids

such as qualitative simulation, providing intermediate steps such as subgoals, and many others. Students should begin working in heavily supported environments and should graduate to successively less heavily supported environments until they are practicing in an environment close or identical to the actual task environment.

6. Review. Old material should be reviewed when new material is introduced. Review functions not only to space practice but also to show students how to discriminate the situations appropriate for the application of old and new skill.

7. The Trials Effect. Skill increases with the amount of practice.

Summary

Before turning to the implementation of maintenance training in interactive media, it is worthwhile to summarize the above discussion. Our review of the cognitive foundations of skilled maintenance reveals three distinct instructional objectives.

Students should be become adept at reasoning from a **mental model** of the equipment being maintained. Training in this aspect of maintenance can be based on a qualitative simulation of the equipment. Appropriate exercises can teach students how the device functions and how its operating procedures can be cast in terms of the mental model. Other exercises can teach students to make inferences and predictions from the structure of the model.

Students must learn to execute **procedures** from memory, from job aids or by actually creating the procedures themselves. Each of these cases requires somewhat different instructional approaches. In general, procedures are learned through practice. They should be presented and practiced in such a way that their structure is clear and that the constraints of structure on function are evident. When a job aid is involved, students should learn procedures for using the aid. Case studies are generally used for instruction in procedures that are derived from the student's problem-solving activities.

Instruction in **troubleshooting** consists of a sequence of problems in which the student must isolate the fault(s) in a device or simulated device. Problems should be chosen in such a way as to promote the acquisition of knowledge-based and context-independent skills. Overly complex problems should be avoided. Problems should be included that address mission-critical faults and their close relatives. Also included should be problems that span the possible structural patterns of the device. The environment for troubleshooting practice should be one that makes evident and promotes the reasoning processes needed for successful troubleshooting. Thus, it should provide for explicit tracking of hypotheses and for commentary on potential troubleshooting actions.

Computer-Based Maintenance Training

Our comments thus far have been largely free of any mention of instructional media. The traditional media for maintenance training have been lecture and simulation. More recently, computers and interactive video have played a role. In this part we discuss the possibilities for computer-based, interactive maintenance training. Our first concern is with automation of the training itself. We then turn our attention to automation of the instructional development process.

Automating the Training Process

Interactive maintenance training (e.g., Towne, 1986) has generally been restricted to high-fidelity simulators. That is, simulators that physically resemble the equipment being maintained. This physical resemblance is implemented either with three-dimensional mockups, with film, and more recently, with video. Instruction usually consists of pure troubleshooting exercises, other aspects of maintenance training being covered by more traditional media.

Recent developments in instructional technology and the design philosophy described above suggest a different approach to interactive maintenance training systems. This new approach would differ from existing methods in several respects.

1. Because of the primacy of mental models in effective maintenance, qualitative simulation with an explicit representation of a mental model would play a primary role in maintenance training.

2. The scope of interactive instruction could be expanded to include instruction in reasoning from a mental model and procedure training (in addition to troubleshooting training).

3. Functions such as critiques and coaching now often removed from the practice context would be incorporated into that context.

4. Depictions of actual equipment (using video, sound, and other means) would be used for the explicit purpose of associating elements of the mental model with their real-world counterparts.

The remainder of this part contains some suggestions for implementing this approach.

The Infrastructure of Interactive Maintenance Training

A strong recommendation of the approach advocated here is the development of an *infrastructure* representing the knowledge to be acquired during training. Instructional methods for conveying this knowledge employ the infrastructure as a source of instructional material. We define the infrastructure's main components in this part and make some suggestions for its instructional use in the next part.

1. Qualitative Simulation. By qualitative simulation, I mean a presentation of a device's function and structure in terms of the kind of mental model described above in "Mental Models of Equipment." This presentation could be implemented using a program such as the Intelligent Maintenance Training Simulator (IMTS) (Towne and Munro, 1988; Towne *et al.*, 1990) where the mechanics of the mental model form the basis of the simulation program itself. Alternatively, the presentation could be based on some other computational approach such as the mathematical model of a steam plant used in STEAMER (Hollan, Hutchins, and Weitzman, 1984). What is important for our purposes is that the simulation appear to the student in the form of a mental model. This means that:

a. the topology of the device is reflected in the display by showing the connections among components;

b. the conceptual structure of the device should be reflected in the simulation by appropriate grouping of systems and subsystems;

c. the state of each component can be made evident to the student through the use of color, icons, or other mechanisms;

d. changes in each components inputs and outputs can be made evident to the student;

e. the student should have full access to the model through simulated controls, indicators, test points, and replacement operations; and

f. provision should be made for the instructional system to configure the simulation in any normal or faulted mode.

2. Physical Simulation. As is mentioned above in the subsections entitled "Mental Models of Equipment" and "Orientation to the Equipment," maintenance concepts and procedures have imaginal as well as conceptual aspects. The turn to qualitative simulation does not, therefore, imply that training should not treat the physical characteristics of the equipment or procedures. Computer-based training treats these

characteristics using video, sound, and other such devices. Needed, therefore, in addition to a qualitative simulation that reflects the conceptual structure of the equipment is a *physical simulation* that reflects the actual appearance of the equipment. Two considerations drive the design of this physical simulation.

First, physical depictions of the equipment must be tied to corresponding qualitative depictions. That is, each component should be shown using a symbolic display (such as an icon) in the qualitative simulation, and an image. Observations and manipulations of the qualitative simulation should have corresponding illustrations of the physical equipment itself.

Second, just as the conceptual structure of the equipment is reflected in the qualitative simulation, its physical structure should be represented in its physical simulation. Views of the equipment's main assemblies and subassemblies should be constructed to reflect the access paths to particular components. The means for simulating assembly and disassembly of the equipment should be provided.

3. Conveying Functionality. Along with structural and physical aspects of maintenance training, we have pointed out the importance of information relating the function of devices. Technicians can rely on functional information—about the function of the device itself and about the function of its subsystems and components—in most or all of the reasoning tasks required for maintenance.

Functional knowledge of a device, its subsystems, and its components is used primarily in maintenance to determine whether or not the device, subsystem, or component is functioning properly. This implies that available within the qualitative simulation should be a characterization of the function of each element so that students can be asked or informed about the functional status of the element. Such a characterization could describe how the system functions under normal operating conditions, indicate the range of acceptable outputs, and contain contextual information such as the elements with immediate connection to the one in question.

For other reasoning tasks, the design or goal structure of the equipment may be of use. If for example, a subsystem cannot be restored to a fully operational state, the maintainer can use information on the purpose of the subsystem to decide on the most effective partial repair. Material should be incorporated into the training system that indicates the role of each component and subsystem in the goal structure(s) employing that component or subsystem.

4. Representing Procedures. Another component in the infrastructure of automated maintenance training is the

computational representation of maintenance procedures. From the discussion in "Cognitive Components of Maintenance Skills - Procedures" above, we can derive the following elements:

a. a description of the procedure's control structure, that is, its steps and choice points,

b. how the procedure employs and manipulates the mental and physical models of the device, and

c. the functionality or goal structure of the procedure.

It almost goes without saying that the representation should be executable. That is, an interpreter should be constructed that can execute the procedure in conjunction with a particular configuration of the mental and physical model.

Any number of formalisms can be used to satisfy these requirements. Perhaps the leading contenders are ATN grammars, production systems, and and-or graphs. The use of one formalism does not exclude others, and in some cases mixtures, combinations, and redundant representations may be useful.

5. Troubleshooting Expertise. Effective troubleshooting practice in both automated and traditional environments depends critically on the availability of troubleshooting expertise. For the purposes of automated training, this troubleshooting expert should reflect the nature of human troubleshooting skills. Thus something on the order of Hunt and Rouse's (1984) fuzzy rule-based model would be appropriate. The essential features of this model are rules that implement both knowledge-based and context-free troubleshooting methods and an explicit representation of the course of the troubleshooting process. Thus, the model could be used to exhibit each step in troubleshooting, its rationale for taking the step, and changes to the problem state after the step has been taken. Rouse and Hunt's model suffers, along with others, from a psychologically unrealistic model of the evaluation of the utility of alternative observations and a simplified mental model of the equipment. It nonetheless contains the major features of an instructionally useful model for troubleshooting expertise.

Instructional Methods

With at least a vague conception of the infrastructure for interactive maintenance training, we can be more specific about the instructional paradigms and curricula. A top-level approach to curriculum design might divide the course into three parts, corresponding to the three main instructional objectives described in earlier subsections entitled "Cognitive Components of Maintenance Skills" and "Instruction," and also, incidentally, reflecting the structure of more traditional maintenance training.

Some specifics of each major course part are given below. In treating each part, we briefly describe some of the exercises that might be used, and we provide guidelines for curriculum design within each part.

1. Teaching a Mental Model.

a. Physical and Conceptual Structure. Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections.

b. Causal Reasoning. Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

c. Functional Reasoning (a). Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

d. Functional Reasoning (b). Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted.

e. Physical and Conceptual Appearance. Students are asked to discriminate among component states on the basis of some physical depiction of those states.

The exact sequence of these exercises should be designed to reflect and convey the overall structure of the equipment. In the typical case, where the equipment can be hierarchically decomposed, the exercises can traverse this decomposition in a depth-first fashion so that students learn to reason about a subsystem immediately after learning to reason about each of its components.

These exercises should also be implemented with a view to whole-task training. Many if not all of them could be embedded in mini-troubleshooting problems in order to illustrate the application of qualitative reasoning to troubleshooting.

2. Teaching Procedures.

a. Operation. Students are required to perform certain operational functions using both a physical and conceptual simulator. That is, each step in the procedure must be executed within the physical simulator and the conceptual simulator. For complex procedures the goal structure of the procedure should be tracked during procedure execution.

b. Calibration. Students work with a physical simulation of the device to practice required calibration and adjustment tasks. A conceptual simulation of the system being adjusted or calibrated shows relations among the components involved in the process.

c. Testing. Students are required to carry out fixed testing procedures on a physical simulation of the equipment. A conceptual simulation of the components being tested is used to exhibit or query the student on the states of these components.

d. Access and Disassembly. Students are given the task of gaining access to a particular component. They use a physical simulation of the device to practice the task. A matching conceptual simulation shows which components are accessible at each point in the procedure.

Curriculum design for procedure training is difficult because of subtask relations among procedures often violate the natural coherence relations. The best recommendation to be made in this regard is that subprocedures should be taught before their procedures and that related procedures should be taught in succession. Thus, for example, if a particular test requires disassembly of the equipment, the disassembly procedure should be addressed before the testing procedure. Furthermore, all disassembly procedures should be submitted to a structural analysis that reveals how different branches provide access to different components. The results of this analysis should be reflected in the curriculum so that students are taught how to access (structurally) neighboring components together.

e. Procedure Selection and Use of Job Aids. Students are asked to identify the procedures needed to deal with particular situations and to select any appropriate job aids. Support is provided for this exercise in the form of subgoals and intermediate steps needed to arrive at the proper selection.

Curriculum design for job aids is (or should be) a simpler matter since the curriculum can be designed to reflect the structure of the aids themselves. Even if the aids are not designed systematically, the instructional designer should develop a procedure for selecting the correct aid and design the curriculum to reflect the structure of that procedure.

Considerations pertinent to whole-task training should be given to all (a-e) of the above methods for procedure training. Some of the procedures addressed by this training constitute whole tasks in an of themselves. Others (e.g., disassembly) are enlisted in the service of superordinate tasks. Whole task training can be partially implemented using an apprentice model in which the student observes an automated expert engaged in some difficult task (say troubleshooting) and practices component procedures, testing, for example, as they arise in the course of the task.

f. Redesign and Jury Rigs. Students are provided with conceptual simulations of tasks requiring complete or partial reconstruction of the equipment. For example, students could be required to restore as much functionality as possible with a limited inventory of spare parts or with other constraints on the reconstruction.

Lessons employing exercises of Type f should be arranged to traverse the major systems and subsystems in a systematic (depth first) fashion.

3. Teaching Troubleshooting.

a. Troubleshooting. Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular faults or sets of faults. Other forms of troubleshooting practice can be found in Brown, Burton, and de Kleer (1982).

b. Reverse Troubleshooting. Students are told that a particular component is faulted. They are required to predict the results of certain observations based on this information. Causal reasoning patterns can be elicited or exhibited during the course of these exercises.

c. Case Studies. Students could be given real case studies of intractable troubleshooting problems. Computer support could be provided for collaborative problem solving and for peer and expert critiques of proposed solutions.

A typical troubleshooting curriculum might have the following lessons.

a. A set of reverse troubleshooting and troubleshooting problems that cover the major topological patterns found in the device. Each pattern would be addressed first by reverse troubleshooting exercises and then by troubleshooting exercises.

b. A set of reverse troubleshooting and troubleshooting problems that cover the equipment's mission-critical faults and their nearest neighbor. Students would first reverse troubleshoot each major fault and its neighbor and then troubleshoot the pair.

c. A repetition of Lesson 1 without reverse troubleshooting.

d. A repetition of Lesson 2 without reverse troubleshooting.

e. A mixture of Lessons 3 and 4.

Reverse troubleshooting in this curriculum plays the role of a cognitive support which is gradually faded from the curriculum. Other cognitive supports (e.g. external hypothesis lists) should also be withdrawn in the last lesson.

Instructional Support

In implementing computer-based training it is important not to lose sight of certain critical functions normally provided by instructors in traditional classroom settings. Of particular concern are functions that establish overall goals, motivation, and coherence to the effort. Instruction of this type is related to what Gagné and Merrill (1990) call the *enterprise* addressed by the instruction. Of equal importance are the tutoring and coaching functions whereby instruction is adapted to the moment-to-moment needs of individual students.

1. Enterprise-Oriented Instruction. Traditional classroom methods may play a role in establishing a student's sense of the maintenance enterprise, but in interactive environments, enterprise-oriented instruction should perhaps be viewed more in terms of arts and entertainment (taken seriously) than traditional instructional methods. Thus, mechanisms such as video clips, computer games, and special effects may be appropriate vehicles for conveying the overall significance of maintenance skills.

2. Tutoring in Interactive Environments. Computer based tutors and coaches have been the subject of several research efforts over the past decade or two (Polson and Richardson, 1988; Psotka, Massey, and Mutter, 1988; Sleeman and Brown, 1982; Wenger, 1987). Some tutoring capabilities have been offered in maintenance training (e.g., Brown, Burton, and de Kleer, 1982; Towne and Munro, 1988; Towne *et al.*, 1990), and some general techniques have been developed that might have a place in maintenance training. Some approaches to computer-based tutoring

and coaching (e.g., Anderson, Boyle, and Reiser, 1985) attempt to derive a student's particular understanding (in an information-processing sense) of an exercise (including the nature of impasses and misconceptions) and to direct advice to that particular understanding. Others (e.g., Burton and Brown, 1982) make a more global evaluation and direct advice to the student whenever specific deviations from optimal behavior are observed. Both types are appropriate to the type of maintenance training suggested here. Since the former requires considerable investigation of intermediate states of learning, the latter are more easily implemented.

Automating the Instructional Design Process

The development of interactive maintenance training of the sort described above would, using current programming and authoring technology, be an ambitious undertaking indeed. It is natural, therefore, to ask how much of the instructional development task could be automated. The ideal situation would be a computer program that could create the entire course out of existing documentation and other material. Unfortunately, such a program is now and probably always will be beyond us. Nonetheless, one can see the opportunity for considerable computational assistance in the course of creating interactive maintenance courses of the type considered here. What follows is a treatment of each of the major components of the training system described above with a view to determining which aspects of its development are amenable to automation.

Mental and Physical Simulations

Since the basis of instruction on the approach described here comprises mental and physical representations of the equipment to be maintained, we should first ask what sorts of computational machinery is available for these representations.

Representation of qualitative reasoning structures and practices has received considerable attention in the literature and a number of computational approaches are available for representing mental models. In addition, as was mentioned above, it may be possible to use a quantitative model of the equipment and provide that mathematical model with the conceptual interface of a mental model.

However, a single shortcoming of available mechanisms for representing mental models is the lack of an approach to the problem of chunking. To my knowledge all current approaches either model the equipment as a flat network of components or rely on the model's designer to provide a hierarchical decomposition. Lacking any progress in automatic chunking of mental models, the division of a device into meaningful systems and subsystems will remain the responsibility of the instructional developer.

Computer representation of the physical aspects of a device appear to be less of a problem than representation of its conceptual structure. The maintenance training community has considerable experience with what is known as "2-D simulation," and a number of systems appear to offer adequate power, typically through the use of videodisc and computer graphics. It should be noted that little in the way of knowledge representation is provided with these systems, thus precluding any reasoning about the physical structure of the device.

Perhaps the most complete and interesting effort on modeling of equipment for maintenance training is the IMTS (Towne & Munro, 1988; Towne *et al.*, 1990). Developers using the IMTS can create a qualitative simulation of a device along the lines suggested in the previous subsection entitled "Cognitive Components of Maintenance Skills." In addition, the IMTS provides an interface to an older type of training simulator, the General Maintenance Training Simulator (GMTS) (Towne, 1986), that provides a physical representation of the device under maintenance. Thus, the combination of these two devices offers all the representational power that is needed for a wide range of devices.

Representation of Procedures

Like devices, the representation of procedures has attracted considerable attention in the cognitive science community. As was mentioned above a number of devices are available for representing procedures. The bad news, in connection with maintenance, is that since the range of maintenance procedures is so broad, no one has found it profitable to provide a system devoted to the representation of maintenance procedures. The good news, however, is that the hardest part of developing such a system would probably be design of the knowledge structures that it employs and manipulates. These knowledge structures are nothing more than the mental and physical models, which puts the prospect of a viable system for representing maintenance procedures well within reach.

Building an Automated Troubleshooting Expert

Recall that the automated troubleshooting expert described in this conception of maintenance training (see subsection entitled "The Infrastructure of Interactive Maintenance Training") must possess both context independent and context-specific skills. The context independent skills, by definition, will be common to all maintenance courses and therefore need not concern the developer of any particular course. Knowledge-based, context-specific strategies vary from equipment to equipment and do therefore concern instructional developers. The best approach to deriving knowledge-based troubleshooting skills is a machine learning mechanism that could derive them automatically in simulated troubleshooting exercises. Since the creation of such a mechanism would be a major research project in its own right, for practical purposes, knowledge-based methods would have to be formulated by subject matter experts for each equipment.

Exercise and Curriculum Development

Computers show considerable promise as devices for generating curricula of exercises and examples. The instructional approach described above relies heavily on such curricula, so that it behooves us to ask how computers might assist in their generation. Needed are

1. a *template* for the curriculum itself or part of the curriculum to be generated by the computer,
2. *frames* for representing exercises in such a way that they can be fit to the template, and
3. a *search mechanism* for filling the template with particular exercises.

In the absence of precise specifications for any of the lessons suggested above, curriculum templates would embody the curriculum suggestions in the above subsection entitled "Instructional Methods" using a special-purpose programming language. The frames for representing individual exercises might contain slots for content, procedure, prerequisites, subtasks, and cognitive support. The search mechanism is essentially an implementation problem and need not concern us here.

Instructional Support

Recall from the previous subsection entitled "Automating the Training Process - Instructional Support" our concern with two areas of instructional support for the training activities suggested here: enterprise-oriented instruction that provides coherence, motivation, and a sense of the overall significance of the maintenance task; and tutorial functions that provide advice appropriate to a student's moment-to-moment situation during the course of an exercise.

The automated generation of enterprise-oriented instruction is, to my mind, about as feasible as the automated generation of academy-award movies or best-selling novels. Thus, any system that serves as a vehicle for maintenance training should provide for the inclusion of human-generated materials of the sort needed to maintain the cognitive integrity of the enterprise under instruction.

By contrast, it is entirely feasible to provide for some forms of automated tutoring with no extra effort on the instructional developers part. A case in point is the IMTS, which evaluates troubleshooting action against an optimal troubleshooting model so that it can suggest more fruitful approaches at appropriate times. The optimal model used by IMTS suffers from the fact that it is more of a competence model than a performance model, and no attempt is made within IMTS to derive an information-processing account of student actions. However, I see no intrinsic difficulties in improving the tutor along these or any of a number of other lines. Some of these improvements, such as the use of empirical bug catalogs in diagnosing student problems, might entail considerable extra effort on the developer's part. Others, however, such as a more psychologically valid method for choosing troubleshooting actions, would require very little extra effort on the developer's part.

Summary

We can summarize the foregoing discussion by describing the minimal set of tasks that would fall to an instructional developer in her interactions with a fully automated instructional design assistant for maintenance training.

The designer would provide a complete, formal description of the device. This description would consist of:

1. models of each type of component used in the device,
2. a matrix of connections among the components,
3. a structural breakdown of the device into its constituent systems and subsystems, and
4. a functional breakdown of the device denoting the purpose of each element.

In addition the developer would provide the materials needed for a physical model of the device, including,

1. a physical breakdown of the device denoting overviews, detail views, and sub-assemblies,
2. imagery (in appropriate media) for depicting all views of the device and its elements in all possible states.

Also needed for development of the infrastructure would be a formal representation of all maintenance procedures addressed in the course of instruction. This representation would denote the control structure of the data so as to permit the simulation of any procedure in the context of a mental and physical models described above. Also represented would be the goal structure of the procedure in a way that would show the relation of function to structure.

A final component of the infrastructure is the troubleshooting expert. As was mentioned above, its context-independent methods are not course specific, but the developer would need to supply the rules characterizing knowledge-based troubleshooting methods for the equipment addressed in each course.

With these infrastructure specifications, the automated design aid would create fully functional mental and physical models of the system. These models could be used to carry out causal reasoning and to simulate the execution of any maintenance action or procedure.

To fill the curriculum, the instructional developer would need to write templates for the generation of exercises in each lesson and would probably need to edit the results of the generation procedure. She would also need to provide any ancillary enterprise-oriented material such as video clips and special effects. Finally, for some tutoring techniques, she might need to provide a catalog of common student misconceptions or mind bugs.

Conclusions

Summary

This discussion has been concerned with the major issues to be faced in the development of interactive maintenance training. A summary of those issues and our conclusions may help to orient us to any overall lessons to be found in the discussion.

What is maintenance?

Maintenance is largely the collection of procedures needed to operate, adjust, test, disassemble (and reassemble), and repair equipment. Of particular importance are the specialized tasks that constitute troubleshooting.

What knowledge and skills support proficiency in maintenance tasks?

A major source of support for maintenance skills is a mental model of the equipment to be maintained. This model describes the equipments subsystems and components, its topology, its functionality, and, through imagery, the physical manifestations of these concepts.

Most maintenance skills are procedural in nature. Like other procedures, maintenance procedures have a functional structure, a control structure, and perceptual-motor (input-output) components. Maintenance procedures can be learned; they can be interpreted from job aids; or they can be created as the result of problem-solving activities.

The specialized skills required for fault isolation rely partly on context-independent reasoning from the device's topology and partly on knowledge-based methods or associations between patterns of observations with troubleshooting actions. In addition, skilled troubleshooters tend to chose the most information-laden actions.

<u>What is required in the way of instruction for maintenance</u>?

Students need first to be taught how to reason from a mental model of the equipment. This is best done through reasoning exercises conducted within appropriate conceptual and physical representations of the equipment.

Maintenance procedures are best taught through structured practice. The practice regimes should make clear any intermediate steps, the appropriate handling of choice points, and the constraints imposed by function on the control structure of the procedure. Where job aids are employed on the job, procedures for using these aids should be developed and taught to students. Problem-solving exercises should be included to address procedures that students must invent on the job.

Troubleshooting is best taught through selected fault isolation exercises. These exercises, in order to be instructive, should be restricted to simple cases. Of particular interest are mission-critical faults, their close neighbors, and faults that illustrate the basic patterns of context-free fault-isolation methods. Practice in troubleshooting should be supported by exhibiting intermediate cognitive steps such a tracking hypotheses about possible faults.

The curriculum structure of all exercises should be governed by known principles of curriculum design. Relevant considerations include the need for a lesson structure evident to the student, the prerequisite structure of the curriculum, its part-whole structure, the need to cover all important cases, the role of cognitive support during learning, the role of review, and the effects of repeated practice.

<u>How can maintenance training be implemented with interactive media</u>?

Two components are needed for maintenance training with interactive media: an infrastructure representing the knowledge to be conveyed, and an instructional system for conveying that knowledge.

The major components of the infrastructure are a computational form of the mental model of the equipment, a corresponding physical model describing and depicting the equipment's physical characteristics, a functional model of the equipment describing its design rationale, a computational representation of all maintenance procedures to be taught, and an automated troubleshooting expert.

These tools can be used to provide instruction addressing the cognitive objectives of maintenance training. Exercises in qualitative reasoning can be used to teach students the mental model of the equipment, its physical manifestation, and its relation to the equipment's function. Procedure practice can be provided in the context of both the mental and physical models so that students can induce the mental structure of each procedure and its physical concomitants. Troubleshooting practice can be provided within the context of the mental model in such a way as to exercise the major skills involved in both context-free and knowledge-based troubleshooting.

Special care must be taken to provide instructional support for the exercises provided by the system. Two forms of support are of particular importance. Enterprise-oriented material should be created to establish the motivation, coherence, and significance of the course as a whole. In addition, an automated tutor should be available to address students' ongoing instructional needs during the course of instruction.

What aspects of instructional development can be automated?

As a programming effort, the creation of a system of the type described here is overwhelming. However, development tools either exist or could be created to automate many of the development chores. Among these tools are those for creating mental and physical models of the equipment, for representing procedures, and for building troubleshooting experts. Tools for automatic creation of instruction are not available at this time, however one can envision systems that would automatically generate a number of the exercises required working from a template or program for the curriculum. Recent work on intelligent tutoring systems could be used to automate or partially automate the development of tutoring and coaching facilities.

Maintenance Training in Perspective

We have, heretofore, treated maintenance training as if it were only of interest in and of itself. It is therefore fitting to ask if any lessons emerge from this discussion about maintenance training as one example of a training development problem or domain. Two observations on this topic strike me as being particularly important.

One of these observations concerns the uniqueness of maintenance as a subject matter domain. Is maintenance a set of highly specialized skills or is it simply a collection of skills with no unique psychological properties. If the above speculations are correct, the answer lies somewhere between these two extremes.

Maintenance does have some unique characteristics. For one thing, there is the pervasive support that a mental model brings to virtually all maintenance activities. While it is true that other endeavors are also supported by mental models, those employed in maintenance are of a unique sort. The particular form of mental models of equipment gives the instructional developer a considerable advantage in defining these models for training purposes. Also unique to maintenance is the special nature of troubleshooting. Context independent strategies for troubleshooting are employed in many different situations, and the form, at least, of knowledge-based strategies is constrained by the nature of the troubleshooting task.

On the other hand, the many procedures that make up maintenance tasks—procedures for repair, testing, etc — do not seem to possess a common structure that is unique to maintenance. The classification of a procedure as one used in maintenance is almost completely uninformative as to its structure, its implementation, or appropriate instructional strategies. All that we have said about procedure training throughout this part would apply equally well (or poorly) to any other domain.

A second observation concerns the focus of effort in instructional development for maintenance training. One cannot help but be struck by the amount of attention that must be given to the infrastructure containing the knowledge to be taught. It appears that this infrastructure is a powerful tool that enables the facile construction of a number of different instructional methods. It is knowledge, one way or another, that supports skilled performance. Hence, it should be no surprise that representing knowledge is a key element in skills training. The foundational position of knowledge in the training system supports the contention that successful instructional development will depend more on developing instructional methods within the framework imposed by a common infrastructure than on developing a number of infrastructures that must conform to a common set of instructional methods. What gives us hope for the automated development of maintenance training are the commonalities of knowledge representation across different applications.

V. TEACHING TROUBLESHOOTING PROCEDURES (Halff)

Introduction

This section is an extension of the previous section for automating the design and delivery of maintenance training. The previous section gave a general characterization of maintenance tasks, the instructional objectives of maintenance training, maintenance training methods, and computer-based maintenance training. My focus in that section was on three aspects of computer-based maintenance training.

First, I suggested that a mental model of the equipment constituted a fundamental objective of maintenance training. Such a model represented effective knowledge of the device including its behavior, structure, function, and appearance. Essential to interactive maintenance training is a computer representation of a mental model of the equipment. This computer representation should consist of a qualitative simulation of the equipment that supports reasoning about the behavior of the equipment on the basis of component models and device topology, and a physical simulation that provides experience with physical observation and manipulation of the equipment.

Second, I suggested that all maintenance procedures to be learned be given a formal representation that makes explicit their control structure and function. These representations should be tied to the qualitative and physical simulations and should be available for guided practice in the procedures.

Third, I proposed that recent findings concerning the nature of skilled troubleshooting be incorporated into troubleshooting training. My interest in this issue was limited to cases in which the technician is required to formulate a troubleshooting strategy based on device knowledge and general troubleshooting principles. In those cases, which I call *troubleshooting as problem solving*, skilled technicians tend to apply context-free rules to local topological patterns found in the device, use device-specific symptom-action associations, and choose information-laden trouble-shooting actions. Of particular importance to training is the possibility (indeed the reality) of incorporating these results into a computer-based expert troubleshooting model.

The proposals in the previous section concerning all three issues suggested that an instructional infrastructure could be built that would permit the easy implementation of a number of computer-based instructional methods for maintenance training. In the last part of that section, I outlined several such methods and provided suggestions for assembling them into maintenance-training curricula.

This section extends the ideas of the preceding in two respects.

First, it focuses on procedure learning and, in particular, on learning a class of standard troubleshooting procedures based on fault trees. This class of methods reduces troubleshooting tasks from the problem-solving activity described in the previous section to one of implementing a general procedure to deal with particular malfunctions. The data needed to implement the procedure for a particular malfunction can be found in the troubleshooting sections of the equipment's technical documentation.

Second, this section is more specific in that it examines, in some detail, the training methods and requirements for one particular troubleshooting procedure, namely, that of repairing an Air Force T-38A aircraft that fails to start on the ground. This procedure is described in USAF Series T-38A Aircraft Organizational Maintenance (1989b). The training proposed below should teach technicians a version of the procedure that is quite close to the one described in the above manual. In what follows, we will refer to the procedure as the *no-start troubleshooting procedure*.

The instructional design approach adopted by the previous section and by most other systematic instructional designs is also reflected in this section. I first describe the content to be taught, organizing the discussion around the no-start troubleshooting procedure. I then derive a set of instructional objectives. Finally, relying on recommendations from the previous section, I draw out the major implications of our analysis for instructional design and practice.

Instructional Content

The T-38A Starting System

Since the malfunction used as an example here is that of a T-38A that fails to start, understanding the troubleshooting task obviously requires some understanding of the starting system of the T-38A. The following is a brief characterization of this starting system taken from various sources (Kieras, 1988; USAF, 1989a, 1989b, 1990). Since I refer to these sources with some frequency in the following, I have adopted, for the sake of readability, the practices of calling USAF Series T-38A and AT-38B Flight Manual (1989a) the "Flight Manual", USAF Series T-38A Aircraft Organizational Maintenance (1989b) the "Engine Conditioning Manual", and to USAF Series T-38A Aircraft Organizational Maintenance (1990) the "Electrical Systems Manual."

The T-38A is a two-engine jet aircraft. Each engine must be started independently, and the right engine is started first. In our discussion of the no-start troubleshooting procedure and related issues, we will assume that the procedure addresses the right engine. The treatment presented here would need some modification for extension to a two-engine version.

Figure 3 is a grossly oversimplified schematic view of the starting system of the T-38A. The reader is cautioned that the picture presented in Figure 3 is far from complete and even inaccurate in many ways. Not shown, for example, are most of the systems servicing the left engine. In addition, components such as the fuel pump and generator, that only function after the engine has started, are not illustrated in Figure 3. Those familiar with the aircraft will also note other inaccuracies where I have deliberately simplified the system or inadvertently made some incorrect assumption about the aircraft. Figure 3 does, however, contain every component that figures significantly in the no-fault troubleshooting procedure, and it represents my best assessment of the interconnections among these components.

Needed to start an engine are compressed **air** (used to *air-motor* or rotate the engine), **fuel**, and **ignition**.

On the ground, air is supplied by an external compressor attached through an air hose to a *diverter valve* on the aircraft. The diverter valve is positioned by the ground crew to direct air to the engine (left or right) being started. It is interesting to note that an earlier version of the aircraft, described in the Electrical Systems Manual and in Kieras (1988) had a mechanism for automatically positioning this valve. Kieras pointed out possible design problems with this automatic mechanism, but we will resist the temptation to speculate on any connection between his writing and subsequent modification of the aircraft.

Fuel for each engine is supplied through fuel lines controlled by the *throttle*. An additional control over fuel flow is an *overspeed governor* that is subject to leaks which can impede the fuel needed for a normal start. A *boost pump*, designed to supplement gravity feed at high altitudes, can be used to purge air from the fuel line.

Ignition for each engine is supplied through two *igniters*, a *main engine igniter* and an *afterburner (AB) igniter*. Each of these igniters provides a train of sparks at a steady rate (3 sparks every 2 seconds).

The delivery of air, fuel, and ignition to the engine takes place under partial control of an electrical starting system, the main elements of which are shown in Figure 3. The pilot starts the engine by depressing a *starter switch* and advancing the throttle to its IDLE position. Depressing the starter switch engages a *timer and holding relay* that, in turn, arms an ignition circuit for about 30 seconds. Once the circuit is armed, moving the throttle to IDLE closes a *throttle cutoff switch* and thereby causes the igniters to fire. Advancing the throttle also delivers fuel to the engine.

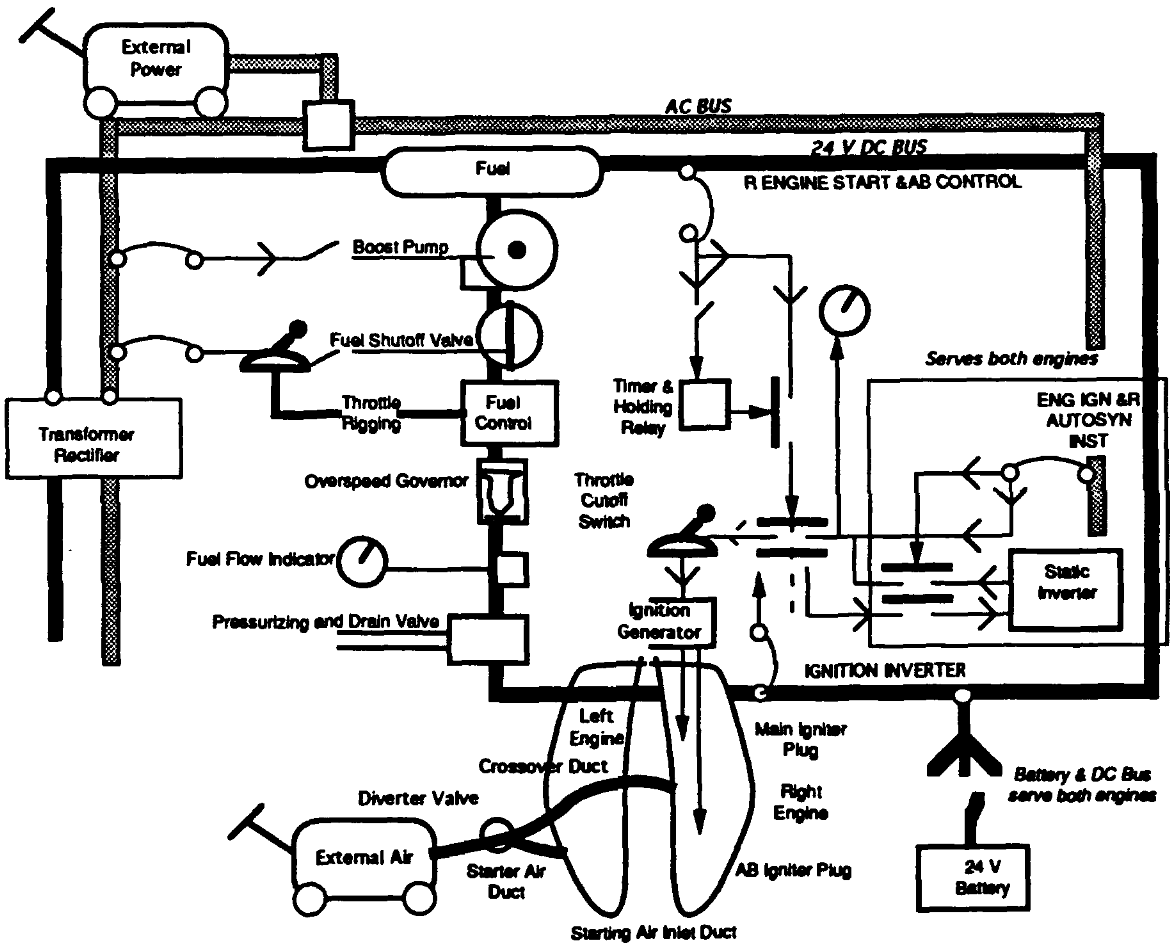


Figure 3. Gross Simplification of the Components Involved in the No-Start Troubleshooting Procedures.

A final important element in the starting system is the delivery of electrical power to the igniters and control circuits. DC Power is needed for the ignition control circuit (i.e., the timer and holding relay); AC power is needed to fire the ignition. On the ground, power can be supplied by two sources: a (DC) battery in the aircraft and an external AC power source. The battery is controlled by a switch which must be turned off if external power is used. If the aircraft is started under battery power, a *static inverter* is used to convert this power to AC for the igniters. If the battery is not used to supply DC power, a *transformer-rectifier* on the aircraft converts AC power to DC. Also of importance to troubleshooting is the fact that the static-inverter circuit powers a subset of the right-engine cockpit instruments and, in particular, the right-engine fuel/oxygen indicator. After starting, electricity (AC) is supplied by generators (not shown in Figure 3) powered by each engine. These generators begin to operate when the engines reach a set cut-in speed.

Starting the T-38A

With this background, the starting procedure for the T-38A should be clear. To paraphrase the Flight Manual (pp 2-6),

1. Make sure nothing is in the way of the aircraft.

2. Set the diverter to the right engine.

3. Apply compressed air to rotate the engine.

4. When the engine speed reaches 14% RPM (or at least 12% RPM), push the right starter switch.

5. Advance the throttle to idle.

6. Wait for the engine to start.

   Ignition should occur before fuel flow reaches 360 lb/hr. If not, turn the throttle off, maintain air flow for 2 minutes to evacuate fuel from the engine, and restart.

The EGT (engine thrust) should begin to rise within 12 seconds of start of fuel flow. If it does not, abort the start.

   The generator should cut in before the 30 sec. ignition circuit times out. If this does not occur check to make sure that the engine light is Normal. If it is, push the start button again to provide electrical power for the start.

7. Check the engine instruments, the hydraulic power, and the caution light panel.

8. Repeat steps 2-7 for the left engine.

9. Disconnect the air supply and, if connected, the external power.

10. Make sure the battery is on.

Possible Faults

Any of a vast number of faults could cause the T-38A engine not to start. Our analysis, however is limited to the small number of faults actually described in the Engine Conditioning Manual. These faults are listed in Table 1. The reader will note that in many cases, our identification of a fault is not all that specific. What constitutes a fault for us is any determination that terminates the no-start troubleshooting procedure. By the same token, what appears in the Repair column of Table 1 is, in some instances a repair, and, in others, directions for further troubleshooting. The limited time and information available to us constrained us to work within the limits of the information provided in the previously identified manuals.

The Troubleshooting Procedure

A careful reading of the Engine Conditioning Manual reveals a critical feature of the procedure described therein, namely, its conformance to a general troubleshooting procedure schema. This schema depends on a representation of the system under maintenance as a hierarchical *fault tree*. Figure 4 shows the fault tree for the no-start troubleshooting procedure. Although Figure 4 is not a perfect match to the description in the Engine Conditioning Manual, the differences between the two have no significance for this discussion. Notice that the terminals of the tree correspond to the faults identified in Table 1. Also notice that some of the terminals are marked as "Last Resort." In the troubleshooting procedure, each such faults is assumed as a last resort when all other faults on its branch have been eliminated.

The fault tree is nothing more than a description of the troubleshooting procedure's structure. Needed to convert that description into an actual procedure is a fault-tree interpretation procedure or schema. Table 2 contains a description of this schema in pseudocode. A more colloquial description is as follows. To troubleshoot any component, first check its overall functionality. If the component is nonfunctional then either repair it, if it is a terminal in the fault tree, or troubleshoot its subcomponents. However, any

Fault	Description	Repair
	Ignition Faults	
Ignition Circuit Breakers Not Engaged.	Three circuit breakers are present in the ignition control circuitry: R ENGINE START & AB CONTROL, ENGINE IGNITION & R AUTOSYN INST, and IGNITION INVERTER. Any or all of these could be improperly engaged.	
Defective Static Inverter.	The static inverter can fail to supply AC power to the igniters.	Replace the static inverter.
Defective Electrical System.	The engine's electrical system can be faulted in such a way that even external AC power is not being delivered to the igniters.	Troubleshoot the electrical system using procedures defined in the Electrical Systems Manual.
Defective Igniters.	The igniters themselves may not fire, even when properly powered.	Remove the engine.
	Fuel System Faults	
Defective Aircraft Throttle Rigging.	The throttle rigging connects the throttle to the fuel control system. If defective, fuel flow may not be adequate for starting.	
Fuel System Circuit Breakers Not Engaged.	The fuel control system will not operate if its circuit breakers are not properly engaged.	
Air in System.	Air in the fuel system may retard fuel flow.	Apply external power and start engine with boost pumps on.
Altitude Problem.	Fuel feed may not achieve required pressure at high altitudes.	
Fuel Shutoff Valve Closed.	A fuel shutoff valve must be open to permit the flow of fuel.	
Excessive Drain from Engine Components.	Fuel may be draining off through engine drain lines.	

Table 1. Faults Causing No Start Conditions

last-resort fault (that is, one isolated solely by eliminating other possibilities) should be repaired without further ado. As is mentioned below, these last-resort faults may simply be guards to prevent the schema from arriving at the impasse of Line 2.3.2.3 in Table 2.

Fault	Description	Repair
Internal Leakage from Engine Drain Indicator.	Excessive amounts of fuel are drained off through the bypass hose of the overspeed governor .	Replace the overspeed governor.
Unknown Fuel System Fault.	Fuel may not flow for reasons other than those listed above.	Remove engine.
Defective Ignition Time-Delay Relay.	Relay is not engaged for 30 sec upon depressing starter, thus preventing electrical power from reaching the igniters.	
Defective Diverter Valve.	The diverter valve is not positioned properly.	
Blocked Air Duct.	The hose from the compressor to the aircraft, may be blocked or kinked, thus preventing compressed air from reaching the engine.	
Defective Engine Starting Air Inlet Duct and/or Crossover Duct.	Ducts from the diverter to the engines may not allow passage of air.	
Engine Seized or Binding.	The engine may not be rotating freely.	Remove engine and determine cause of problem.
	Operational Problems	
Altitude Problem.	Engine may not start at high altitudes.	

Table 1. (Concluded)

A critical aspect of this fault-tree approach is the structure of the tree itself. In particular, the structure of the tree (a) reflects the structure of the equipment itself as a hierarchy of subsystems and (b) is predicated, by virtue of Line 2.1 on the availability of tests of the functions of all components except last resorts.

Needed to instantiate the schema for the no-start troubleshooting procedure are Table 1, Figure 4, and the observations needed to perform the tests called for in Line 2.1. These observation procedures, are listed in Table 3.

To summarize, this section describes the no-start troubleshooting procedure of the T-38A as a fault tree. Needed to fully specify the procedure are:

```
To troubleshoot a component

 1.0 If the component is a Last Resort then
 1.1 Repair the component
 1.2 Conclude* that the component was faulted
  2.0 Else {component is not a last resort}
    2.1 Check the overall functionality of the component
    2.2 If the component is OK then
       2.2.1 Conclude that the component is not faulted
     2.3 Else {component is not OK}
       2.3.1 If the component has no subcomponents
         2.3.1.1 Repair the component
         2.3.1.2 Conclude that the component was faulted
       2.3.2 Else {component has subcomponents}
         2.3.2.1 For each subcomponent of the component
           2.3.2.1.1 Troubleshoot the subcomponent
           2.3.2.1.2 If the subcomponent was faulted then
             2.3.2.1.2.1 Conclude component was faulted
           2.3.2.1.3 End If
         2.3.2.2 End For
         2.3.2.3 Conclude fault is unknown
       2.3.3 End if
     2.4 End if
   3.0 End If

Note:*"Conclude" statements are equivalent to function
returns in that they terminate the troubleshooting
procedure and return a value describing the fault thus
isolated.
```

Table 2. Fault-Tree Interpretation Schema

1. the fault tree indicating components and subcomponents that can be faulted together with their order of testing (Figure 4),

2. procedures used to test the functionality of each component in the fault tree (Table 3),

3. repair procedures for each terminal component in the tree (Table 1), and

4. a procedure for interpreting these data and thereby executing the troubleshooting procedure (Table 2).

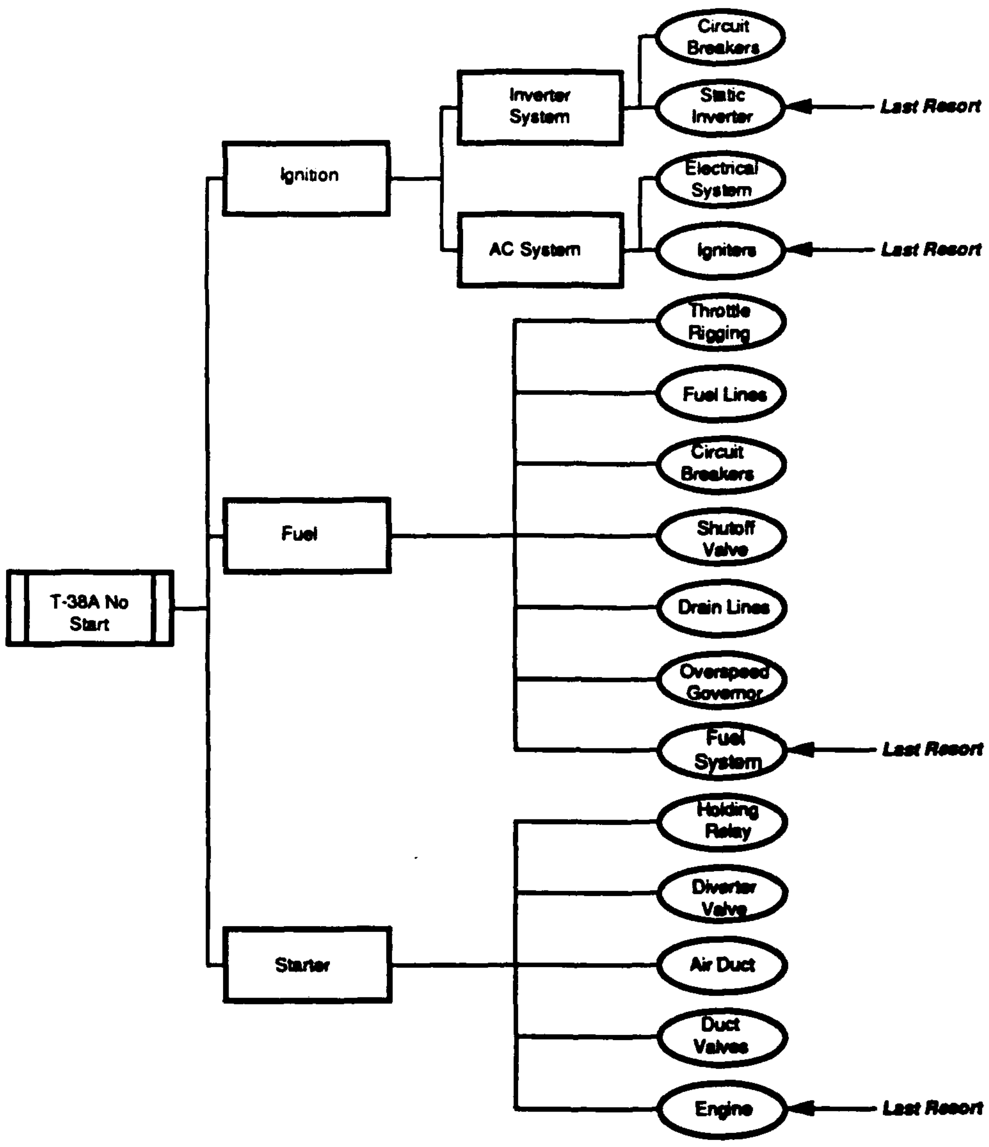


Figure 4. T-38A No-Start Fault Tree.

An engaged reader should, at this point, have several objections to the entire approach suggested here. For one thing, examination of the procedure itself, as defined in the Engine Conditioning Manual is far from optimal. It calls for the repetition of some observations, for movement from the cockpit to the tailpipe and back again and for some observations that appear to yield no information whatsoever.[5]

[5]Kieras, Personal Communication, October 1990.

Component	Test
T-38A No Start	Standard starting procedure
Ignition	Observe AB and main plug firing during ignition without external power
Inverter System	Check Fuel/Oxy Indicator without external power
Circuit Breakers	Check engagement
Static Inverter	Last Resort[a]
AC System	Observe AB and main plug with external power
Electrical System	Check voltage at Pin N in engine ignition and accessories disconnect plug[b]
Igniters	Last Resort
Fuel	Check for fuel mist in exhaust
Throttle rigging	Advance throttle past idle and check for fuel flow or ignition
Fuel lines	Attempt repair by running boost pumps to clear air
Circuit breakers	Check for engagement
Shutoff Valve	Observe status
Engine Drain Line	Check for excessive fuel drainage
Overspeed governor	Check flow rate through bypass hose
Fuel System	Last Resort
Starter	Engine achieves 14% RPM with proper air flow
Holding Relay	Check engagement for 30 sec
Diverter valve	Inspect position
Air Duct	Check for kinks, obstruction
Duct valves	Check position
Engine	Last resort

Note: [a]Last-Resort components are always deemed to be faulted by a process of elimination.
[b]Not shown in Figure 1.

Table 3. Functionality Tests for Fault Tree Components

One may also ask why the Air Force, having adopted a fault-tree approach, chose not to present the procedure in those terms in the Engine Conditioning Manual. Or, perhaps more sensibly, one could ask why I chose to present the procedure as a fault tree.

The answer to all of these questions, or at least the first and the last, is that a fault-tree representation of a troubleshooting procedure offers cognitive benefits that outweigh any inefficiency that it may introduce into the troubleshooting process. Without belaboring the point, a fault tree is a hierarchical structure typical of those found in most highly developed cognitive skills to simplify overly complex procedures and thus render them amenable skilled execution (Chase and Ericsson, 1982). Using more efficient but less structured approaches would inevitably result in slower learning and in lower terminal speed and accuracy of performance.

I can only speculate as to why the fault-tree structure is not explicit in the Engine Conditioning Manual. One possibility is that the authors assume that this structure will be evident to technicians using this volume. Another is that the direct presentation of the procedure is well suited to execution if learning is not a concern. A third possibility is that the Engine Conditioning Manual was written by a skilled technician who had long before automated the procedure to the extent that the fault-tree no longer played a role in its execution.

Another serious issue is that of the generality of the approach. The above description is based on the analysis of a single troubleshooting procedure. I have not investigated any other documented procedures, even for the same aircraft. Nor have I verified that the procedure described in the Engine Conditioning Manual bears any resemblance to actual troubleshooting. Thus, the conclusions and recommendations based on this analysis have little in the way of empirical support.

Despite these objections, the instructional design proposed below is predicated on the assumption that a fault-tree representation of the no-fault troubleshooting procedure (and other procedures of the same sort) is the most appropriate representation for instructional purposes. It therefore forms the basis for the instructional objectives described below.

Instructional Objective

Since much of this effort relies on the general scheme described in the previous section, we need to frame the particular content described above in terms of the general objectives described in that section. Recall that these instructional objectives consist of a mental model of the equipment, the procedures used to maintain the equipment, and the special skills needed for troubleshooting as problem solving.

Mental Models

Mental models of equipment, as described in the previous section have three aspects: structure, function, and imagery.

Device Structure

The structure of a device is a formal qualitative description of that device, often called a *qualitative model*. Such a model describes the behavior of individual components, that is, how they change state in response to changes in input and how their outputs vary according to state. It also describes the behavior of the device as a whole, that is, how the outputs of one device are connected to the inputs of another.

Fault trees of the type described above are closely related to qualitative models of equipment in that the qualitative model therefore provides cognitive support for the troubleshooting procedure. For example, inspection of the right engine fuel/oxygen indicator for evidence that the static inverter system is functioning correctly only makes sense within the context framed by a mental model of the ignition system as shown in Figure 3.

A qualitative model of the entire starting system could be (indeed has been)[6] created from a conception like that shown in Figure 3 and this figure would make a reasonable basis for a qualitative model if our only concern was the no-start troubleshooting procedure. However, any realistic maintenance course must also address many other malfunctions and troubleshooting procedures. The model shown in Figure 3 is not really a system but rather the components of several systems that, taken together, are responsible for starting the aircraft or failing to start it. A different function (e.g., establishing radio communications) or even a different malfunction of the same system (e.g., failure to start in flight) would involve different components of the same or different systems. If troubleshooting training were to be based on models like that of Figure 3, a different model would be needed for each malfunction in the curriculum. The proliferation of such models would easily defeat any advantage of model-based training.

Needed, therefore, is one or a small number of models that can support troubleshooting training for all malfunctions of interest. Function or malfunction-specific models like that of Figure 3 may still have a place in training and may even take a measure of cognitive reality. However, a more general scheme is needed to define the mental models that constitute instructional objectives in this context. The design of such a scheme for the T-38A is beyond the scope of this section. Nonetheless, we can describe a strategy for constructing a mental model of the aircraft. This strategy has a bottom-up and a top-down component. The bottom-up component calls for a fault-tree analysis of each malfunction covered in the Engine Conditioning Manual, the Electrical Systems Manual, and other such documents. One result of this analysis will be a collection of tables like Table 1 and Table 3, specifying components and elementary procedures for repair and observation. This information can be

[6]Towne, personal communication. October 1990.

used (along with other technical documentation) to construct component models that reflect normal and faulted behavior and that can be appropriately manipulated during troubleshooting.

The top-down component of the strategy calls for analysis of the aircraft's systems using the scheme laid out in Section I of the Flight Manual. As illustrated in Figure 5, this scheme describes the major systems of the aircraft: fuel, fuel control, electrical, etc., together with their subsystems.

Combining the bottom-up and top-down analyses is a matter of merging the primarily component-wise information in the former with the topological and structural information in the latter.

Device Function

For purposes of troubleshooting, device function is best thought of in terms of operating functions and malfunctions. Section VI of the Engine Conditioning Manual is an excellent starting point for such an analysis. Identified in that section (entitled "Troubleshooting Engine and Related Problems") are a number of operating modes of the aircraft. For each operating mode is a list of malfunctions. For each malfunction, a troubleshooting procedure (theoretically, based on a fault tree) is provided which identifies particular components and subcomponents. Thus, it is possible to construct a functional hierarchy based on operations, potential malfunctions, and the normal or faulted operation of individual components. This hierarchy for T-38A engine functions is shown in Figure 6.

This functional hierarchy is not the same as the structural breakdown by system shown in Figure 5, particularly since several systems are usually involved in each operation. Starting Operation, for example involves the starter system, the fuel system, the fuel control system, the ignition system, the electrical power system, and the engine. The functional hierarchy of Figure 6 is organized around the T-38s functions or missions, and it comprises a distinct, important aspect of the instructional objectives. Below, in the following subsection entitled "System Function", we will discuss the part that it plays in curriculum design.

Imagery

Imagery, as conceived of in the previous section, is the perceptual face of a technician's conception of the equipment. As such, it covers the appearance (in sight, sound, and other senses) of the equipment, the physical actions that implement observations and repairs, and knowledge of the physical location of components and subsystems.

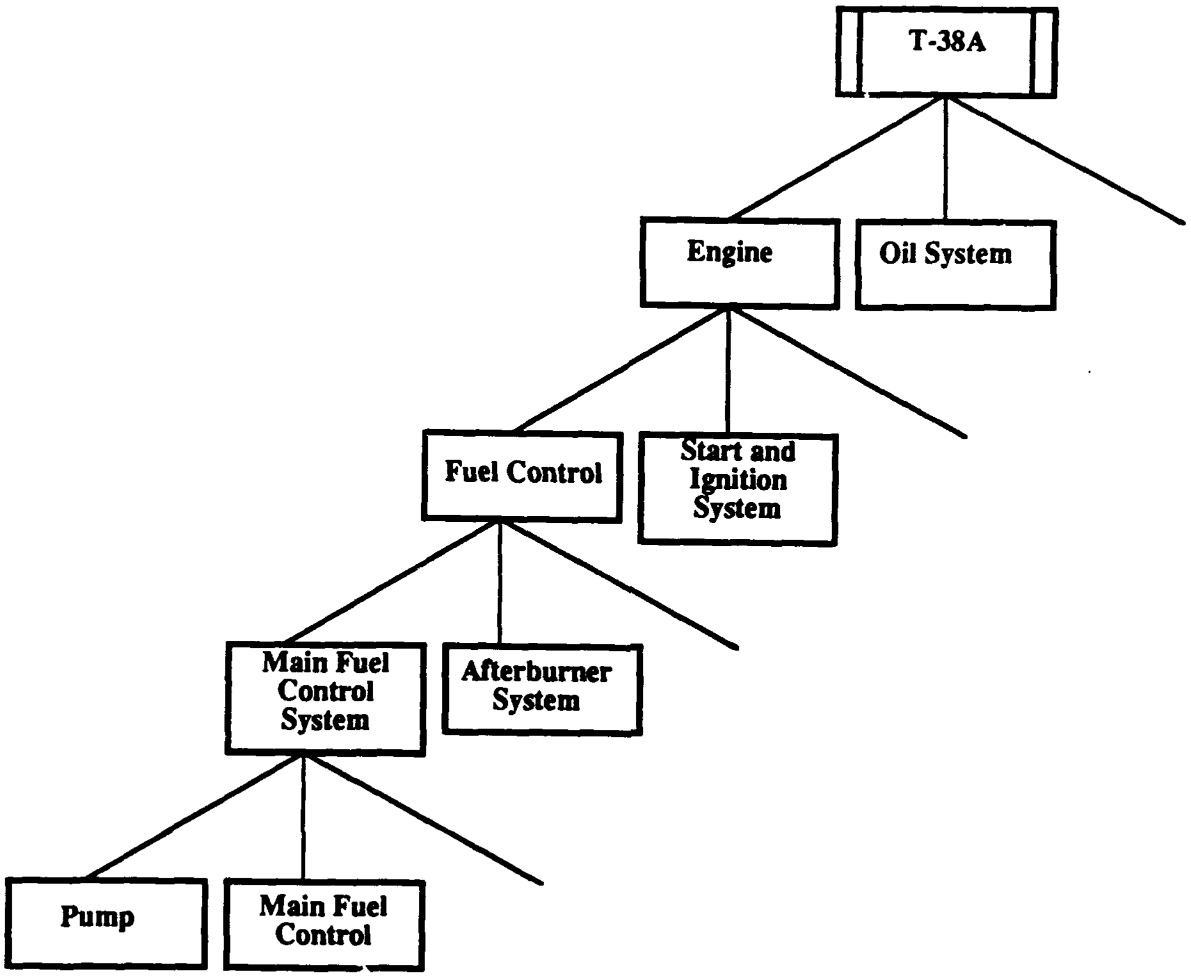


Figure 5. Structural Breakdown of the T-38A

Direct formal representation of the imaginal aspects of a mental model are well beyond current methods of knowledge representation. What can be constructed, however, is a system of tokens that stand for different perceptual chunks. These tokens would include, in the case at hand,

1. the perceptual aspects of all observations, such as the sound of the main igniter sparking and the appearance of the fuel-flow indicator;

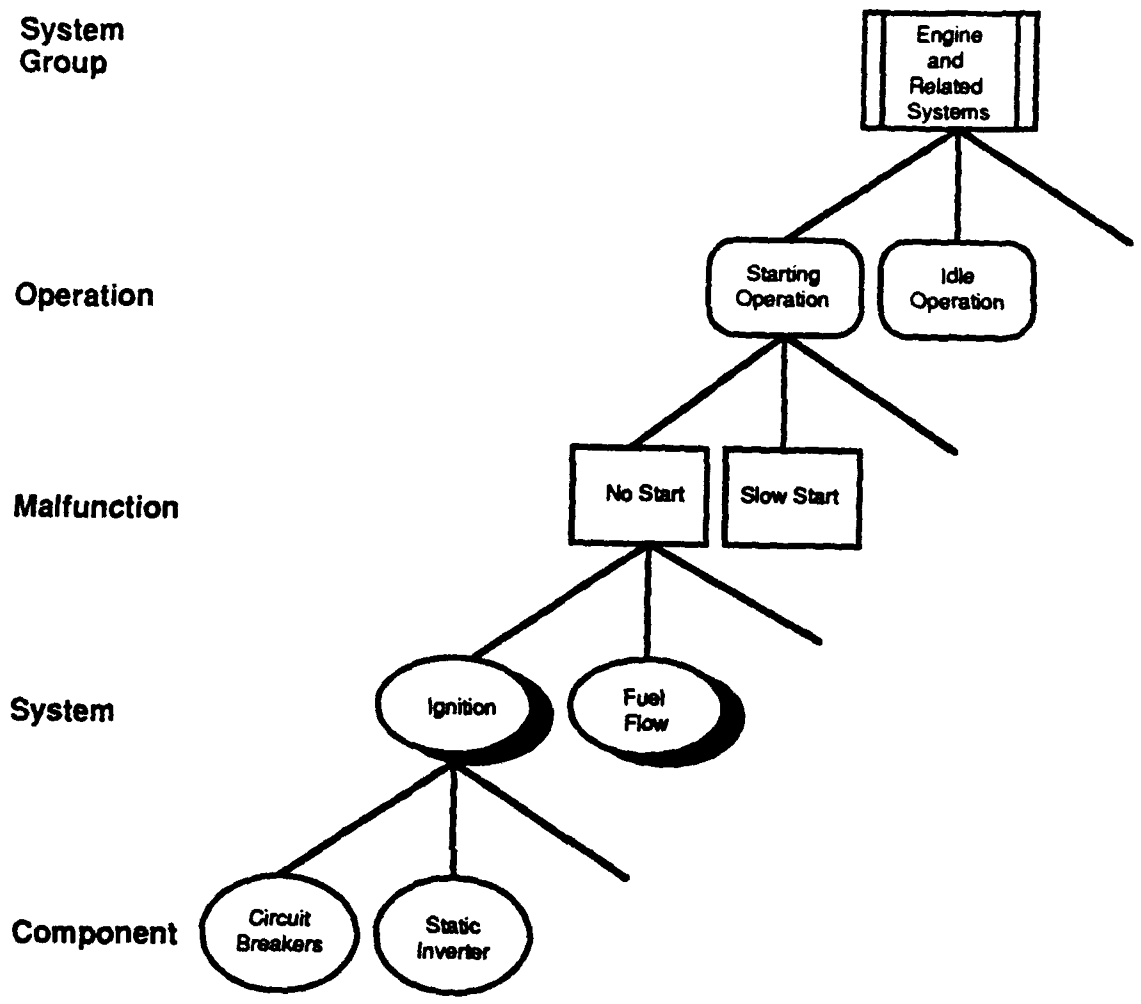


Figure 6. Functional Breakdown of T-38A Engine-Related Components

2. the actions needed to effect operations and repairs, such as shorting out the afterburner igniter and engaging a circuit breaker; and

3. transitional items needed to refocus attention from one component or operation to another, for example, locating the ignition-system circuit breakers or moving from the cockpit to the rear of the aircraft.

The Troubleshooting Procedure

The second type of instructional objective proposed in the previous section is that of procedural knowledge. Some aspects of procedural knowledge representation are discussed in that section, but a more informed description can be found in Section II. To a large extent, both of these general discussions are superseded by the specific description of troubleshooting in the previous subsection entitled "Starting the T-38A."

This subsection suggests that three distinct items of procedural knowledge are involved in mastering the no-start troubleshooting procedure:

1. the general troubleshooting schema of Table 2,

2. the testing and repair procedures listed in Table 1 and Table 3, and

3. the structural information in the fault tree itself (Figure 4).

These items constitute three, almost independent, procedural objectives for instruction in the no-fault troubleshooting procedure. By "almost independent" I mean first that the fault-tree interpretation schema can be learned in such a way that it will apply to any fault tree, not just the one presented in Figure 4. Also, the testing and repair procedures can be mastered outside of no-fault troubleshooting procedure and most will probably be useful in other maintenance tasks. Mastery of the fault tree (Figure 4) does, however, depend on mastery of the fault-tree interpretation schema.

Representation mechanisms for the types of procedural knowledge of concern here are well discussed in Section II. Recommended there is the use of Card, Moran, and Newell's (1983) GOMS scheme for representing procedural knowledge. Representational schemes such as GOMS constitute important but not complete tools for expressing fault-tree based troubleshooting procedures.

Representing the Fault-Tree Interpretation Schema

The fault-tree interpretation schema of Table 2 is basically a control structure wrapped around retrieval operations that fetch the appropriate elementary repair and operation procedures. The control structure in this schema can be translated in a straightforward manner directly into GOMS. Not specified in Table 2, however, are the operations needed to retrieve information that instantiates the schema. For example, Line 2.1 requires retrieval of a procedure of the sort listed in Table 1.

Line 2.3.1 requires the retrieval of structural information of the type specified in Figure 4. These retrieval operations will vary according to situation and skill level of the technician. Novices will use technical documentation, instructors and other external sources. Intermediate technicians will retrieve this information from long-term memory. In highly-skilled technicians, the schema and its instantiating data will be compiled into a single procedure that resembles the description in the Engine Conditioning Manual. A GOMS representation of the schema in Table 2 must therefore provide different strategies for retrieval operations to reflect the circumstances and skill levels of the technician.

Representing Elementary Procedures

The elementary procedures listed in Table 1 and Table 3 are eminently suited for representation as GOMS structures. Indeed, a preliminary GOMS analysis of the entire no-start troubleshooting procedure is available and is provided here in Appendix A. Much of the material in this appendix are analyses of the operations in Table 1 and Table 3.

It is worth noting in passing that Appendix A offers a reasonable representation of the procedure used by a highly skilled technician. Control knowledge and operational knowledge are compiled into one procedure and the fault tree is given no explicit representation at all. Most of the literature on skill development indicates that this composition or compilation is a characteristic of highly automated skills.

Representing the Fault Tree

Fault trees such as that shown in Figure 4 are, on the face of it, declarative knowledge, and, in students, they may begin life in declarative form. To be of use, however, their fundamental representation must be procedural. The procedural requirements of the interpretation schema (Table 2) are the following functions.

1. Retrieve the procedure for checking the functionality of a component (Table 2, Line 2.1).

2. Retrieve the procedure for repairing a terminal component (Table 2, Lines 1.1 and 2.3.1.1).

3. Decide whether or not a component is terminal (Table 2, Line 2.3.1).

4. Retrieve successive subcomponents of a non-terminal component (Table 2, Line 2.3.2.1).

A minimal representation of a fault tree must therefore implement these four functions. For instructional purposes a declarative representation will also be needed together with procedures (perhaps GOMS procedures) for implementing these functions by interpreting the declarative representation.

Troubleshooting as Problem Solving

The last category of instructional objective mentioned in Section IV is that of troubleshooting. My concern in including that category was for troubleshooting as a **problem-solving** activity in which the technicians need to discover the appropriate fault isolation strategy. In particular, my reading of the literature in this area suggested a problem solving procedure with three main components:

1. context free rules for isolating faults based on topological patterns,

2. device-specific rules for troubleshooting based on known symptom-fault associations, and

3. procedures for choosing the most information-laden actions at particular choice points.

This problem-solving approach to troubleshooting relies only on connectivity information in the mental model and a set of device-specific observation-action associations.

By contrast, the focus of this section is on troubleshooting **procedures**, in which the isolation strategy is provided through documentation and instruction and need only be implemented by the technician.

Troubleshooting as problem solving is an important aspect of skilled troubleshooting and therefore of maintenance training, even when a large body of troubleshooting procedures is available. In a system of any complexity, no troubleshooting guide offers complete coverage of the possible faults, and seldom will a guide offer procedures for dealing with complex situations such as test-equipment unreliability, multiple interdependent faults, and intermittent faults.

Schemes are available for representing troubleshooting as problem solving (Hunt and Rouse, 1984; Towne, Johnson, and Corwin, 1983), and instructional systems are available for implementing these schemes. Although these representation schemes could be improved (see Section IV) this section is not the place to focus on the issue.

Of central concern, however, is the relationship between the procedural approach to troubleshooting presented in the previous subsection entitled "The Troubleshooting Procedure" and problem-solving approaches such as those of Hunt and Rouse (1984) and Towne, Johnson, and Corwin (1983). The fault-tree approach, by sacrificing generality, provides the structure needed to render the troubleshooting process manageable with limited time and cognitive resources. Methods for troubleshooting as problem solving have greater generality but are expensive in terms of both time and resources. A combination could use the fault-tree procedure in the initial stages and invoke a problem solving process to deal with cases not covered in the fault tree.

Earlier we mentioned that the last-resort faults in Figure 4 seemed to be guards against the impasse that might occur if a component malfunctions but each of the tested subcomponents functions properly (see Line 2.3.2.3 of Table 2). In addition to guarding the procedure against impasses, these last-resort faults may also play the role of place holders that mark occasions for troubleshooting as problem solving. If so, then the fault tree could be redesigned to eliminate the last-resort faults, and an impasse itself (i.e., Line 2.3.2.3) could signal the occasion for invoking a problem solving process.

The implications of this suggestion for curriculum design are clear. That is, troubleshooting as problem solving should be taught after, and in conjunction with procedures based on fault trees, and students should be explicitly taught when to enter a problem-solving mode of troubleshooting. These points of transition are the Last Resort items in Figure 4 or, alternatively the impasses marked by Line 2.3.2.3 of Table 2.

Summary

The suggestions made above regarding instructional objectives for the no-start troubleshooting procedure parallel the general suggestions in Section IV.

A mental model should be developed that covers the structural, functional, and imaginal aspects of the T-38A. The structural aspects can be represented using any of a number of schemes for qualitative modeling. The model thus derived should cover all components germane to documented T-38A troubleshooting procedures and should reflect the overall structure of the aircraft as described in Section I of the Flight Manual. A functional breakdown, based on operations, function, and malfunctions (see Figure 6) should describe the function of each component and provide a method for discriminating between normal and malfunctioning operation. The imaginal aspects of the mental model are best implemented as tokens or placeholders that refer to perceptions, actions, and transitions in attention.

The no-start troubleshooting procedure, and others of its ilk, should be represented using a fault-tree scheme. This scheme is based on a general fault-tree interpretation procedure described in Table 2 and cast so that it accommodates the different information-retrieval methods appropriate for different situations and skill levels. Also part of the scheme is the set of elementary procedures for observation (Table 3) and repair (Table 1). A third aspect of the fault-tree approach is a representation of the fault tree itself in procedural and declarative terms. Its procedural representation should include the functions needed for retrieval of structural information in the interpretation schema of Table 2.

We also made brief mention of problem-solving approaches to troubleshooting that select local troubleshooting moves or actions based on device topology, known observation-action associations, and the information value of potential actions. We suggested that students need to be taught these problem-solving techniques and should be trained to invoke them when the fault-tree approach arrives at an impasse.

Instructional Material and Methods

Section IV proposes certain methods for using computers in maintenance training. The discussion in that section, and in this one as well, is organized around three aspects of a computer-based instructional system.

Infrastructure

An instructional infrastructure for computer-based maintenance training consists of computational representations of the instructional objectives. In particular, it contains mechanisms that can simulate and describe the equipment under maintenance in both qualitative and physical terms. It also contains mechanisms that can interpret and describe procedures to be learned. In addition, an expert troubleshooting system should be available for teaching troubleshooting as problem solving.

Qualitative and Physical Simulation

Recall that the mental model consists of a qualitative model of device behavior, a description of device function, and a collection of tokens that stand for imaginal (perceptual) aspects of the device. The qualitative model can be implemented in a computer using available qualitative simulation systems such as that described in Towne, Munro, Pizzini, Surmon, Coller, and Wogulis (1990). These systems have the following basic capabilities.

1. They can instantiate any qualitative model of the sort described in Section IV.

2. They can carry out qualitative reasoning on the model and thereby determine the states and outputs of specified components based on information regarding states and state-changes in the rest of the device.

3. They present to the student a graphical depiction of the system being simulated. This depiction does not generally resemble the physical appearance of the system. Rather, it represents components as icons that can be manipulated by the student and whose appearance reflects the states of the corresponding components. A display from such a system might, for example, bear a passing resemblance to Figure 3.

4. They provide different views of the system. Presented to the student not just as one view of all components, but rather a collection of views, each showing only the components selected for that view. Some of these views might show the behavior of major subsystems, and thus conform to the breakdown illustrated in Figure 5. Others, might conform to the functional breakdown illustrated in Figure 6 in that, like Figure 3, they show all of the components involved in a particular function.

Constructing a qualitative model of the T-38A, then is a matter of entering the qualitative model of the aircraft into a qualitative simulation system, designing the graphical representation of the model, and constructing the views needed for instruction. For troubleshooting training, three types of views are needed.

1. Global subsystem views are needed that present each major subsystem as an independent unit.

2. Views like that in Figure 3 should be constructed to show all of the components involved in the functions and malfunctions selected for troubleshooting.

3. A third type of view conjoins the first two and thereby depicts only the components involved in particular lower level nodes of the fault tree. Figure 7, for example, provides a view of the Fuel System branch of the fault tree in Figure 4.

Needed to support the imaginal aspects of the model is a physical (as opposed to qualitative) simulation of the aircraft. This physical simulation presents to the student, through appropriate media, the sights, sounds, and manipulanda associated with each of the tokens in the imaginal model (see previous subsection entitled "Imagery"). These requirements include:

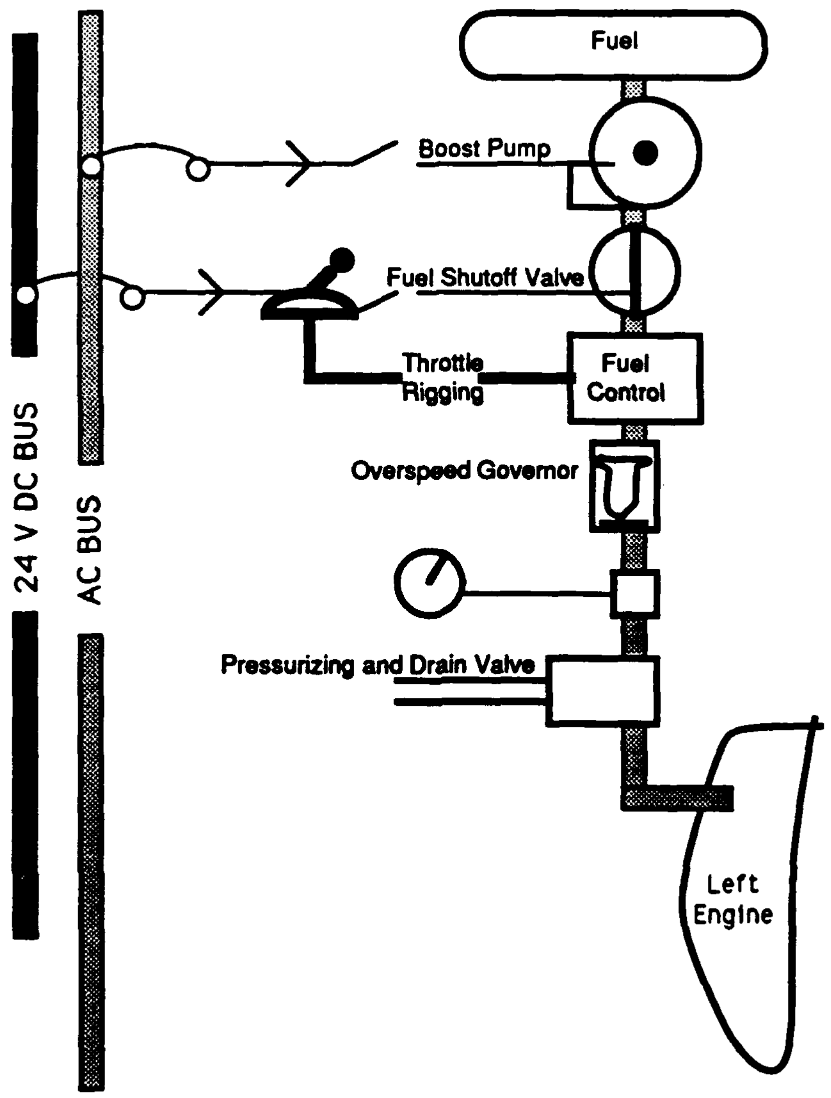


Figure 7. Components Involved in the Fuel Flow Branch of the No-Start Fault Tree (Figure 4)

1. a simulation of all observations that might be made in the course of troubleshooting, including indicators such as the Right Fuel/Oxygen Indicator and other sorts of observations such as the appearance of fuel mist in the exhaust;

2. a simulation of all actions that might be undertaken in the course of troubleshooting, including controls such as the throttle and other actions such as that of shorting out the AB ignition plug; and

3. a simulation of transitions that refocus attention from one component of the aircraft to another, including views of panels such as the instrument panels and gross changes of view such as descending from the cockpit and moving to the rear of the aircraft.

As a development strategy, the team developing instruction should walk through every branch of every troubleshooting procedure, noting the views and manipulations needed to support the procedure. These notes can then form the basis for production of the physical simulation in appropriate media.

Naturally, the physical and qualitative simulations should be linked in their implementation so that manipulations of the physical simulation are manifest as state changes in the qualitative simulation, and manipulations in the qualitative simulation are manifest in displays of the physical simulation. For example, if the student, in the qualitative model closes the fuel shutoff valve, then a view of the exhaust in the physical simulation (with the throttle on) should reveal no fuel mist. Conversely, if the student, in the physical simulation, applies external power to the aircraft, then the Ignition Power Transfer Relay[7] should be shown to energize in the qualitative simulation.

Procedure Interpretation

A second component of the instructional infrastructure is a computational implementation of the procedures needed for, in our case, troubleshooting. Recall that the formalism used for representing these procedures is the fault tree described in the previous subsections entitled "The Troubleshooting Procedure." The components of this formalism are

1. a fault-tree interpretation schema, represented in GOMS or some similar formalism,

2. the fault-tree itself, represented in both procedural and declarative forms, and

3. the observation and repair procedures attached to nodes in the fault tree, also represented in GOMS or some other appropriate formalism.

Needed to computerize these formal models are an interpreter that can execute the troubleshooting procedure, appropriate links

[7]This relay, shown but not labeled in Figure 3, is energized by the AC bus. When engaged, it directs power from the AC bus to the ignition circuit and disconnects the AC output of the static inverter from this circuit.

to the simulation models discussed above, and a means of presenting the structure of the procedures themselves to the student. The first two items pose no particular challenges since interpreters are available for this purpose and since the simulations themselves possess the mechanisms needed to link them to a procedure interpreter.

The third item, the means of presenting the procedure itself to the student, is a critical aspect of the instructional design proposed here. Mechanisms are needed for explicit presentation of each of the three parts of the fault-tree approach.

Presenting the Fault-Tree Interpretation Schema. Support for navigating the fault-tree interpretation schema of Table 2 should be provided by a display that either indicates the major milestones in the schema or allows the student to indicate these milestones. These milestones include:

1. checking the functionality of the component under consideration (Table 2, Line 2.1),

2. isolation and repair of a component (Table 2, Line 2.3.1),

3. troubleshooting the subcomponents of a component (Table 2, Line 2.3.2.1), and

4. dealing with unsolved cases (Table 2, Line 2.3.2.3).[8]

We can distinguish several ways of presenting milestone transitions to students, depending on the level of guidance required.

1. In heavily guided practice, the computer should be capable of dictating the next milestone to be reached in the procedure. For example, "We have just determined that no fuel is flowing. We will examine each of the potential causes of this problem in turn."

2. In more relaxed guidance, the student should be required to indicate the next milestone. For example, "We have just determined that no fuel is flowing. What do we do next in the fault tree?"

3. Under even more relaxed guidance, the computer might simply note milestones as they occur and check to ensure that the student's actions are consistent with the

[8]For basic instructional purposes, students should be protected from these impasses. However, impasses can be used to give problem-solving practice to more advanced students.

schema. For example: "We have just determined that no fuel is flowing. You are checking the Starter Air Inlet Duct. Re-examine the fault tree to make sure that this is the next step."

Presenting the Structure of the Fault Tree. As with the interpretation schema, both a declarative and procedural interpretation of the tree itself is needed. The declarative presentation is perhaps best done graphically, using a display like that of Figure 4. As the procedure traverses the tree, the active elements can be highlighted in some way and/or subtrees can be displayed as a means of focusing attention (see Figure 8 for examples).

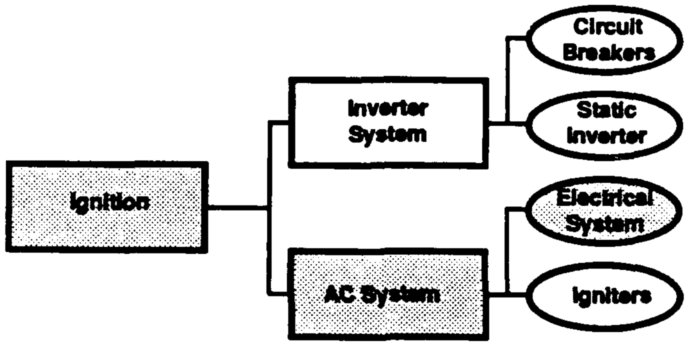


Figure 8. Use of Subtree and Highlighting to Focus Attention

The procedural aspects of fault tree structure are defined as a list of functions in the previous section entitled "Representing the Fault Tree." A computer implementation of these functions should make their results available to students and should be able to query students about the structure of the tree. In particular, the computer should have the means to

1. inform the student of the procedure needed to check the functionality of any component—for example, "To check for low fuel flow, use the procedure that checks fuel mist in the exhaust."

2. ask the student to designate or execute the procedure needed to check the functionality of a component—for

example "Select the procedure for checking the overspeed generator,"

3. inform the student of the procedure for repairing any faulted component—for example, "To repair the static inverter, use the procedure for replacing the static inverter,"

4. ask the student to designate the procedure for repairing a faulted component—for example, "Select the procedure used to repair the disengaged IGNITION INVERTER circuit breaker,"

5. etc.

These functions, can, like those listed in the previous part, be used in guided practice to ensure that the student learns how to properly manipulate the fault tree.

Presenting Procedures for Observation and Repair. Procedures for repair of faulted starting system components are presented in Table 1. Procedures for checking the functionality of components are listed in Table 3. As instructional objectives, both types of procedures are represented in some formalism such as GOMS (see previous subsection entitled "Representing Elementary Procedures").

In a computer-based instructional system, these procedures need to be given a declarative, verbal description, such a plain-English paraphrase of the GOMS representation, and a procedural implementation in the qualitative and physical simulations. In this way students can be shown or asked how the procedure unfolds through interactions conducted in verbal terms, in terms of physical actions and observations, or in qualitative, conceptual terms.

The power of these separate presentation schemes is multiplied by joining them together in complete or partial presentations of the entire troubleshooting procedure. As the procedure is presented and practiced, the system or the student can focus on any of five critical aspects of the procedure:

1. strategy—the status of the procedure with respect to the fault-tree interpretation schema,

2. tactics—the status of the procedure with respect to the fault tree itself,

3. stepwise descriptions—the elementary observation and repair procedures,

4. implementation—the physical actions and observations (in the physical simulation) that implement the procedure, and

5. conceptual aspects—the theoretical description (in the qualitative model) of the procedure.

With the computer implementation suggested above, we view each of these aspects as being potentially available for presentation or practice at any point in the procedure.

Problem Solving and Troubleshooting

Above, in the previous subsection entitled "Troubleshooting as Problem Solving," we discussed troubleshooting as a problem solving activity as opposed to the fault-tree procedural approach described in the previous section entitled "The Troubleshooting Procedure." In Section IV, I suggest that the computational support needed for effective training of these problem-solving skills is a troubleshooting expert based on the problem-solving methods that students should master to solve particular problems.

To my knowledge, two such experts are available, namely, PROFILE (Towne, Johnson, and Corwin, 1983) and the Fuzzy Rule-Based Model of Hunt and Rouse (1984). The former is an almost pure information-theoretic approach that achieves not only efficient troubleshooting but also an impressive match to human troubleshooters. The latter, however, offers more face validity in that it incorporates some of the heuristics known to be used by human troubleshooters.

I will not speculate on which of these models is more appropriate for training in this situation. I will mention, however, that whatever model is used, that model should operate within the context of the qualitative simulation described above. It should also be able to start in mid problem, and in particular, when the fault-tree procedure reaches an impasse. It should also be able to present or explain its choice of each troubleshooting action as problem-solving proceeds.

Instructional Methods

Instructional methods constitute the procedures for engaging the student in instructional interactions. Consistent with the view of instructional objectives described above, Section IV proposes that some of these procedures address the acquisition of a mental model of the equipment. Some should address acquisition of procedures. Some should address the problem-solving skills needed for effective troubleshooting.

The foregoing part has provided us with a powerful set of tools for addressing these goals. To specify the instructional methods completely, we need to configure these tools and assemble them into a curriculum. The discussion above suggests that the curriculum for troubleshooting training should be organized around a few major types of activities that reflect advancing levels of knowledge and skill.

System Behavior and Structure

The first level of understanding to be attained by students is that of a mental model of the device. Activities promoting such attainment include exercises with both the qualitative and physical simulations within a framework of the structural breakdown of the aircraft (Figure 5). Students at this stage will master fundamental qualitative reasoning tasks such as predicting the behavior of individual components and determining the implications certain states of the equipment. Exercises to be used in this phase of the curriculum are listed in Section IV (subsection entitled "Instructional Methods"). To quote from that section,

1. Physical and Conceptual Structure. Students are shown images of the physical equipment and asked to identify individual components, their function, and their immediate connections.

2. Causal Reasoning. Students are given information about all inputs to a component or subsystem and required to predict the state of the component or subsystem, its outputs under normal operating conditions, and its outputs in each possible fault mode.

3. Functional Reasoning (a). Students are shown some of the inputs to an element of the device and asked how its other inputs must be set in order to achieve a desired function or state.

4. Functional Reasoning (b). Students are shown the actual outputs and inputs to an element and asked to determine whether or not the element is faulted.

5. Physical and Conceptual Appearance. Students are asked to discriminate among component states on the basis of some physical depiction of those states.

The exact sequence of these exercises should be designed to reflect and convey the overall structure of the equipment. In the typical case, where the equipment can be hierarchically decomposed, the exercises can traverse this decomposition in a depth-first fashion so that students learn to reason about a subsystem immediately after learning to reason about each of its components.

These exercises should also be implemented with a view to whole-task training. Many if not all of them could be embedded in mini-troubleshooting problems in order to illustrate the application of qualitative reasoning to troubleshooting.

System Function

A second phase of the curriculum should give students an understanding of how components function, or fail to function, together to meet the operational purposes of the aircraft. Put differently, students in this phase should learn

1. what the system and its components are supposed to do,

2. how to make the system fulfill those functions, and

3. how to determine when the system is not meeting its functions.

Activities at this level are organized around the functional breakdown of the aircraft, illustrated schematically in Figure 6. By making this structure evident in instruction, students should be able to induce the basic operations of the system and how its components are involved in those operations.

Activities addressing system function are operational in nature. Some exercises provide guided practice procedures such as starting the engines and in elementary repair and observations of the type listed in Table 1 and Table 3. Practice should begin with the qualitative simulation alone, pass to a stage with joint qualitative and physical simulations, and end with the physical simulation alone. Each subphase should begin with demonstration of the procedure followed by practice.

The order of presentation of different procedures in this phase should conform to a depth-first traversal of the functional breakdown, and, more importantly to the subgoal structure of the procedures themselves. Specifically, the GOMS representation of each procedure breaks the procedure into small manageable subgoals that can be mastered individually. The curriculum should obviously take advantage of this feature of the representation.

A second type of activity in this phase should teach the students the tests or observations needed to check the functionality of each component of the aircraft. Several types of exercises can be used to effect such teaching.

1. Students can participate in qualitative reasoning exercises that address the test procedures. For example, they might be asked to predict the behavior of the Right Fuel/Oxygen Indicator in aircraft that have normal or faulted static inverters.

2. Students can practice observational procedures, such as checking fuel flow in the overspeed governor, in the context of testing the system's functionality. Exercises that place the student in an apprenticeship role can be used to implement this strategy.

3. Students can be asked to both select and execute test or observational procedures. For example, where a beginning student would practice testing the AC bus by being informed that the test involved checking Pin N in the engine ignition and accessories disconnect plug, more advanced students would practice the same operation by simply being instructed to test the AC bus.

By the end of the second phase of training, students should have many of the component skills of fault-tree based troubleshooting procedures. In particular, they should know how to test any of the aircraft's components and subsystems for normal functioning, and they should have a good idea, from exposure to structures like that of Figure 6, of the structure of the fault trees to be used in troubleshooting.

Troubleshooting Procedures

In the third phase, students should be introduced to fault-tree interpretation and to the fault trees that define the set of troubleshooting procedures to be learned. The activities that address these topics are described in some detail above in the subsection entitled "Procedure Interpretation." Arranging these exercises into a curriculum is not a difficult task. The following guidelines seem reasonable in this respect.

1. Every malfunction's fault tree should be covered first in a depth-first traversal of the tree and then in a random order.

2. Initial exercises should provide explicit support for use of the fault-tree interpretation schema (see previous subsection entitled "Procedure Interpretation)." This support can be withdrawn after practice with a few malfunctions.

3. The first exercises with each malfunction should rely heavily on an explicit representation of the fault tree for the malfunction. This support should be faded with advanced practice.

4. Exercises should begin first in the qualitative simulation. As students master the structure of the fault tree, use of the physical simulation can be phased in and support from the qualitative simulation can be faded.

5. Documentation that is normally available in the field should be available on-line.

Problem Solving

We have noted above the importance of teaching troubleshooting as a problem-solving activity. In the previous subsection entitled "Problem Solving and Troubleshooting," I suggested that problem-solving activities have a particular place whenever fault-tree methods arrive at an impasse (Line 2.3.2.3 of Table 2). Although students in the trouble-shooting procedure phase (previous subsection entitled "Troubleshooting Procedures") of the curriculum should be protected from such impasses, students ready for this problem-solving level should be presented with impasses as opportunities to practice problem solving.

Section IV contains proposals for instruction in troubleshooting as problem solving. The following is a list of suggested exercises taken from that section.

1. Troubleshooting. Students are provided with a conceptual simulation containing a single faulted component. At each point in the troubleshooting exercise, students would choose an action and exhibit the consequences of the action. The exercise could take many forms. For example, students might be prompted to select actions diagnostic of a particular faults or sets of faults. Other forms of troubleshooting practice can be found in Brown, Burton, and de Kleer (1982).

2. Reverse Troubleshooting. Students are told that a particular component is faulted. They are required to predict the results of certain observations based on this information. Causal reasoning patterns can be elicited or exhibited during the course of these exercises.

3. Case Studies. Students could be given real case studies of intractable troubleshooting problems. Computer support could be provided for collaborative problem solving and for peer and expert critiques of proposed solutions.

A typical troubleshooting curriculum might have the following lessons.

1. A set of reverse troubleshooting and troubleshooting problems that cover the major topological patterns found in the device. Each pattern would be addressed first by reverse troubleshooting exercises and then by troubleshooting exercises.

2. A set of reverse troubleshooting and troubleshooting problems that cover the equipment's mission-critical faults and their nearest neighbor. Students would first reverse troubleshoot each major fault and its neighbor and then troubleshoot the pair.

3. A repetition of Lesson 1 without reverse troubleshooting.

4. A repetition of Lesson 2 without reverse troubleshooting.

5. A mixture of Lessons 3 and 4.

Reverse troubleshooting in this curriculum plays the role of a cognitive support which is gradually faded from the curriculum. Other cognitive supports (e.g. external hypothesis lists) should also be withdrawn in the last lesson.

What needs to be added to this description is

1. that these exercises should only be introduced in the context of an impasse in the fault-tree procedure, when, for example, all components under "Fuel" in Figure 4 function properly but fuel is still not evident in the exhaust;

2. that the exercises should only be introduced when the student has mastered the fault tree for the malfunction, and

3. that the exercises should be conducted in the presence of a tutor working the problem under the same set of initial conditions as is the student.

Summary

The central contribution of this section is the instructional analysis of troubleshooting procedures based on fault trees. The general approach suggested here to the teaching of fault-tree based troubleshooting can be summarized with the following points.

A fault-tree based troubleshooting procedure consists of a hierarchical fault tree, a general schema for interpreting the tree, and the elementary observation and repair procedures needed to implement the procedure. All three aspects of the procedure need to be mastered by students.

Training should begin, not with the procedure itself, but rather with instruction oriented to the structure and behavior of the equipment and its components. This initial instruction establishes the student's mental model and can be implemented using qualitative and physical simulations of the equipment.

Training on elementary observation and repair procedures should be introduced in conjunction with instruction on device functionality. This instruction should teach students how the equipment is used, how components operate together to achieve the functions of the equipment, and how to determine when the device or any of its components is not functioning properly.

Training on the use of fault trees should be based on guided practice. Initially the interpretation schema and the trees should be made explicit in the instruction. As students advance this explicit support can be withdrawn.

When students are ready to master troubleshooting as problem solving, practice opportunities should be provided within the context of fault-tree procedures. In particular each problem solving exercise should begin by driving the fault tree procedure to an impasse in which a component malfunction cannot be traced to a malfunction in any of its subcomponents.

VI. CONCLUSIONS (Hioki)

Polson & Polson, Polson & Kieras and Halff present interesting discussions on critical components needed to enable automated and intelligent assistance to the novice in the design and development of effective CBI. The specific areas of concern here include automating the cognitive task analysis process and design and delivery of maintenance procedures.

Polson & Polson and Polson & Kieras emphasize the importance of integrating a task analysis function that is "*meaningful and correct*" to support the subsequent instructional design, development and delivery phases. If possible, this function should be automated to the extent possible and include the capabilities to capture subject matter explanation, breadth of component representation and explicit procedures. These capabilities are critical to successful instruction.

Halff continues the discussion by outlining the need for automating critical instructional design elements. He specifically addresses the requirements for representing appropriate mental models, procedures and trouble shooting skills, and their roles as prerequisites in accomplishing maintenance tasks. Following this theoretical discussion, Halff provides an instantiation of this approach with content from an operational aircraft maintenance procedure.

By integrating current advances in the related areas of computer hardware and software technologies, cognitive science and instructional design with the ideas and recommendations of this paper, the outcome could contribute significantly to the initial framework of an AIDA system. Taken together with other theoretical discussions documented in Volumes 1 and 3 of this series may ultimately determine the level of instructional success awaiting future recipients of CBI.

REFERENCES

Anderson, J. R. (1982). Acquisition of cognitive skill. Psychological Review, 89, 369-406.

Anderson, J. R. (1983). The architecture of cognition. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. Psychological Review, 94(2), 192-210.

Anderson, J. R. (1990). Analysis of student performance with the LISP tutor. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. 27-50). Hillsdale, NJ: Erlbaum.

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. Artificial Intelligence, 42, 7-49.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1984). Cognitive principles in the design of computer tutors. In Sixth Annual Conference of the Cognitive Science Society Program (pp. 2-16).

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. Science, 228, 456-462.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. Human Computer Interaction, 5, 1-48.

Brown, J. S., Burton, R. R., & de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems (pp. 227-282). London: Academic Press.

Burton, R. R. & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems (pp. 79-98). London: Academic Press.

Campione, J. C., & Brown, A. L. (1990). Guided learning and transfer: Implications for approaches to assessment. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. 141-172). Hillsdale, NJ: Erlbaum.

Card, S. K., Moran, T., & Newell, A. (1983). The psychology of human computer interaction. Hillsdale, NJ: Erlbaum.

Chase, W. G., & Ericsson, K. A. (1982). Skill and working memory. In G. H. Bower (Ed.), The psychology of learning and motivation (Vol. 16, pp. 1-58). New York: Academic Press.

de Kleer, J., & Brown, J. S. (1983). Assumptions and ambiguities in mechanistic mental models. In D. Gentner & A. L. Stevens (Eds.), Mental models (pp. 155-190). Hillsdale, NJ: Erlbaum.

de Kleer, J., & Brown, J. S. (1985). A qualitative physics based on confluences. In D. G. Bobrow (Ed.), Qualitative reasoning about physical systems (pp. 7-83). Cambridge, MA: Bradford.

Frederiksen, J. R., & White, B. (1990). Intelligent tutors as intelligent testers. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. 1-26). Hillsdale, NJ: Erlbaum.

Frederiksen, N. (1990). Introduction. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. ix). Hillsdale, NJ: Erlbaum.

Gagné, R. M. (1968). Learning hierarchies. Educational Psychologist, 6, 1-9.

Gagné, R. M. (1985). The conditions of learning and theory of instruction. New York: Holt, Rinehart and Winston.

Gagné, R. M., & Merrill, M. D. (1990). Integrative goals for instructional design. Educational Technology Research & Development, 38(1), 23-30.

Glaser, R., & Bassok, M. (1989). Learning theory and the study of instruction. In M. R. Rosenwig & L. W. Porter (Eds.), Annual Review of Psychology, (Vol. 40, pp. 631-666). Palo Alto, CA: Annual Reviews, Inc.

Halff, H. M., & Spector, J. M. (1991). Designing an Advanced Instructional Design Advisor: Possibilities for Automation (Volume 3 of 6) (AL-TP-1991-0008). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory, Human Resources Directorate.

Hollan, J. D., Hutchins, E. L., & Weitzman, L. (1984). STEAMER: An interactive inspectable simulation-based training system. The AI Magazine, 5(2), 15-27.

Hunt, R. M., & Rouse, W. B. (1984). A fuzzy rule-based model of human problem solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-14, 112-120.

Johnson (1988). Developing expert systems knowledge bases in technical training environments. In J. Psotaka, L. D. Massey, & S. A. Mutter (Eds.), Intelligent tutoring systems: Lessons learned (pp 85-111). Hillsdale, NJ: Erlbaum.

Jones, M. K., Li, Z., & Merrill, M. D. (1990a). Knowledge representation for ID_2: Part 1. Logan, UT: Utah State University, Department of Instructional Technology.

Jones, M. K. Li, Z. & Merrill, M. D. (1990b). Knowledge representation for ID^2: Part 2. Logan, UT: Utah State University, Department of Instructional Technology.

Karat, J. (1983). A model of problem solving with incomplete constraint knowledge. Cognitive Psychology, 14, 538-559.

Kieras, D. E. (1982). A model of reader strategy for abstracting main ideas from simple technical prose. Text, 2, 47-82.

Kieras, D. E. (1988a). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), Handbook of human-computer interaction (pp. 135-157). North Holland: Elsevier.

Kieras, D. E. (1988b). What mental model should be taught: Choosing instructional content for complex engineered systems. In J. Psotaka, L. D. Massey, & S. A. Mutter (Eds.), Intelligent tutoring systems: Lessons learned (pp 85-111). Hillsdale, NJ: Erlbaum.

Kieras, D. E. (1990). The role of cognitive simulation models in the development of advanced training and testing systems. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. 51-74). Hillsdale, NJ: Erlbaum.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. Cognitive Science, 8, 255-273.

Kieras, D. E., & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. Journal of Memory and Language, 25, 507-524.

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 365-394.

Kintsch, E., Tennyson, R. D., Gagné, R. M., & Muraida, D. J. (1991). Designing an Advanced Instructional Design Advisor: Principles of Instructional Design (Volume 2 of 6) (AL-TP-1991-0017). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory, Human Resources Directorate.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integrations model. Psychological Review, 95, 163-182.

Kosslyn, S. M. (1980). Image and mind. Cambridge, MA: Harvard University Press.

Lee, A. (1989). Timing of feedback in tutoring systems. (Technical Report No. 89-10), Boulder: University of Colorado at Boulder, Institute of Cognitive Science.

Li, Z., & Merrill, D. M. (1990). Transaction shells: A new approach to courseware authoring. Logan, UT: Department of Instructional Technology, Utah State University.

Nathan, M., Kintsch, W., & Lewis, C. (1988). Tutoring algebra word problems (Tech Report No. 88-12), University of Colorado at Boulder, Institute of Cognitive Science.

Newell, A., & Simon, H. A. (1972). Human problem solving. Englewood Cliffs, NJ: Prentice-Hall.

Olson, J. R., & Olson, G. M. (1990). The growth of cognitive modeling in human-computer interaction since GOMS. Human-Computer Interaction, 5(2&3), 221-265.

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension fostering and comprehension monitoring activities. Cognition & Instruction, 1, 117-175.

Polson, M. C. (in press). Cognitive theory as a basis for instruction. In M. Spector, M. C. Polson, & D. Muraida (Eds.), Automating instructional design: Concepts and issues. Englewood Cliffs, NJ: Educational Technology.

Polson, P. G. (1987). A quantitative theory of human-computer interaction. In J. M. Carroll (Ed.), Interfacing thought: Cognitive aspects of human-computer interaction. Cambridge: Bradford Books/MIT Press.

Polson, P. G., & Kieras, D. E. (1985). A quantitative model of the learning and performance of text editing knowledge. San Francisco: ACM.

Polson, P. G., Muncher, E., & Engelbeck, G. (1986). A test of a common elements theory of transfer. In M. Mantei & P. Orbeton (Eds.), Proceedings CHI'86 Human Factors in Computer Systems (pp. 78-83). New York: Association for Computing Machinery.

Polson, M. C., & Richardson, J. J. (Eds.). (1988). Foundations of intelligent tutoring systems. Hillsdale, NJ: Erlbaum.

Polson, M. C., Tennyson, R. D., & Spector, J. M. (1991). Designing an Advanced Instructional Design Advisor: Cognitive Science Foundations (Volume 1 of 6) (AL-TP-1991-0007). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory, Human Resources Directorate.

Psotaka, J., Massey, L. D,. & Mutter, S. A (Eds.). (1988). Intelligent tutoring systems: Lessons learned. Hillsdale, NJ: Erlbaum.

Regian, J. W., & Schneider, W. (1990). Assessment procedures for predicting and optimizing skill acquisition after extensive practice. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill knowledge acquisition (pp. 297-324). Hillsdale, NJ: Erlbaum.

Reiser, B., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (1989). Knowledge representation and explanation in GIL, an intelligent tutor for programming (Tech Report No. 37), Princeton: Princeton University Cognitive Science Laboratory.

Scardemalia, M. E., Bereiter, C., McLean, R. S., Swallow, J., & Woodruff, E. (1989). Computer-supported intentional learning environments. Journal of Educational Computing Research, 5, 51-68.

Schmalhofer, F., Kuehn, O., Messamer, P., & Charron, R. (1989). An experimental evaluation of different amounts of receptive and exploratory learning in a tutoring system. Behavioral Research Methods, Instruments, and Computers, 22(2), 179-183.

Schneider, W. (1985). Training high-performance skills: Fallacies and guidelines. Human Factors, 27(3), 285-300.

Schneider, W., & Detweiler, M. (1988). The role of practice in dual-task performance: Toward workload modeling in a connectionist/control architecture. Human Factors, 30(5), 539-566.

Simon, H. A. (1975). The functional equivalence of problem solving skills. Cognitive Psychology, 7, 268-288.

Singley, M. K., & Anderson, J. R. (1989). The transfer of cognitive skill. Cambridge, MA: Harvard University Press.

Sleeman, D., & Brown, J. S. (Eds.). (1982). Intelligent tutoring systems. New York: Academic Press.

Spector, J. M. (1990). Designing and Developing an Advanced Instructional Design Advisor (AFHRL-TP-90-52). Brooks AFB, TX: Technical Training Research Division, Armstrong Laboratory, Human Resources Directorate.

Thorndike, E. L. (1914). Psychology of learning. New York: Teachers College.

Thorndike, E. L., & Woodward, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. Psychological Review, 8 ,247-261.

Towne, D. M. (1986). The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), Advances in man-machine systems research (Vol 3). Greenwich, CT: JAI Press.

Towne, D. M., Johnson, M. C., & Corwin, W. H. (1983). A performance-based technique for assessing equipment maintainability (Tech. Report 102). Los Angeles, CA: University of Southern California, Behavioral Technology Laboratories.

Towne, D. M., & Munro, A. (1988). The intelligent maintenance training system. In J. Psotka, L. D. Massey, & S. A. Mutter (Eds.), Intelligent tutoring systems: Lessons learned. (pp. 479-530) Hilldsale, NJ: Erlbaum.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., Coller, L. D., & Wogulis, J. L. (1990). Model-building tools for simulation-based training. Interactive Learning Environments, 1, 33-50.

U. S. Air Force (1989a), USAF Series T-38A and AT-38B Flight Manual (T. O. 1T-38A-1, Change 2).

U. S. Air Force (1989b). USAF Series T-38A Aircraft Organizational Maintenance: Engine Conditioning Technical Manual (T. O. 1T-38A-2-6-2, Change 35).

U. S. Air Force (1990). USAF Series T-38A Aircraft Organizational Maintenance: Electrical Systems Technical Manual (T. O. 1T-38A-2-7, Change 36).

van Dijk, T. A., & Kintsch, W. (1983). Strategies of discourse comprehension. New York: Academic Press.

Wenger, E. (1987). Artificial intelligence and tutoring systems: Computational and cognitive approaches to communication of language. Los Altos, CA: Morgan Kaufmann.

Winograd, T. (1975). Frame representation and the declarative procedural controversy. In D. Bobrow & A. Collins (Eds.), Representation and understanding, New York: Academic Press.

Appendix

T38 Engine Troubleshooting Procedures - Analyst: David Kieras, U. Michigan

Notes

Notation

- o Elementary domain specific operators of airplane controls indicated with*, complex domain specific operators are in {brackets}, otherwise are
- o Steps or Method inferred by analyst are indicated with prefix ?
- o For reference, original procedure numbers shown in [brackets] for each Method or step
- o Notation is based on Kieras (1990) A guide to GOMS task analysis, made somewhat more compact

Note especially the domain-specific operators and the list of Goals for which there are no methods provided in the procedures

- o The user will have to learn how to do these from some other source

Method structure overview

- o Top level Method corresponds to one of the "Trouble" headings - "No Start"
  - Terminates when a second level Method returns that a problem was found
- o Second level Methods corresponds to the "Probable Cause" headings
- o Third level are the step in the procedure, but only roughly, due to the haphazard segmentation of steps in the procedures. Original steps numbers are shown for reference

Analyst decisions

- o Decisions on when trouble shooting is done need to be spelled out
- o When steps subordinated, or grouped by mode, I've shown them hierarchialized

Comments and questions on content of procedures

- o *Other questions and comments listed in italics in the methods*
- o While troubleshooting steps are listed in order of probable cause,
  - Clearly are not lowest cost
    - e.g. listed order has you attempting to start engine separate times
    - e.g. why not do all cockpit checks at once:
      - check all breakers
      - static inverter test with fuel gage
      - check that starter can motor engine adequately, etc.
- o Procedure seems to apply only to right engine start - what if left won't start?
  - Static inverter shouldn't be involved
- o Why is there no test mentioned of the ignition inverter - part of the engine?
- o Step to step flow of control is usually not explicit - no go to step x statements
- o Top level is ambiguous about whether a symptom is observed or is to be looked for
  - e.g. low fuel flow
- o Procedures do not take into account that observations would be made in earlier steps that are not explicitly checked until later
  - e.g. if starter fails to produce 14% rpm
  - e.g. if engine is seized and doesn't rotate
- o Some procedures seem to be dangerous - e.g. looking up tailpipe or check for fuel mist at tailpipe
  - Have some steps been left out?

T38 Engine Troubleshooting Procedures - Analyst: David Kieras, U. Michigan

Method summary

Methods defined for these goals:

1. apply shorting stick to AB plug
2. check AB plug
3. check boost pump pressure at inlet manifold
4. check flow rate from bypass hose
5. check for altitude limitation problem
6. check for bad ignition
7. check for fuel flow problem
8. check for starting system defective
9. check fuel flow indicator
10. check fuel shutoff valve is open
11. check igniter plugs
12. check ignition with external power
13. check jet engine starter
14. check main plug
15. check overspeed governor for internal leakage
16. check starting circuit breakers
17. check static inverter electrical power
18. check throttle rigging
19. determine if fuel flow is low
20. diagnose engine no start symptom
21. obtain shorting stick
22. restore bypass hose
23. set up overspeed governor bypass hose for check
24. try start with boost pumps

Methods invoked for these goals:

1. air-motor engine
2. apply external electrical power to aircraft
3. apply shorting stick to AB plug
4. attach brass or steel wool to end of stick
5. check AB plug
6. check air duct from jet engine starter for blockage or kinking
7. check aircraft electrical system
8. check boost pump pressure at inlet manifold
9. check diverter value for proper positioning during start cycle
10. check engine drain lines for excessively draining component
11. check engine for freedom of rotation
12. check engine starting air inlet duct and crossover duct check valves for freedom of operation and proper installation
13. check flow rate from bypass hose
14. check for altitude limitation problem
15. check for bad ignition
16. check for fuel flow problem
17. check for power at Pin N of engine ignition and accessories disconnect plug
18. check for starting system defective
19. check fuel flow indicator
20. check fuel mist discharge
21. check fuel shutoff valve is open
22. check fuel system circuit breakers
23. check igniter plugs
24. check ignition time-delay relay for 30+/- 3 sec duration
25. check ignition with external power
26. check jet engine starter
27. check main plug
28. check overspeed governor for internal leakage
29. check start circuit breakers

30. check static inverter
31. check static inverter electrical power
32. check throttle rigging
33. determine if fuel flow is low
34. get cap
35. get graduated container
36. get stick and brass or steel wool
37. measure pressure
38. measure quantity of fuel discharged from hose in 1 minute
39. obtain Handbook limitations
40. obtain operating information
41. obtain shorting stick
42. remove engine from aircraft
43. restore bypass hose
44. set up overspeed governor bypass hose for check
45. start engine
46. stop air-motoring
47. try start with boost pumps
48. turn on boost pumps

Methods invoked from more than one method for these goals:

1. air-motor engine
2. start engine
3. apply external electrical power to aircraft
4. check igniter plugs
5. check throttle rigging
6. remove engine from aircraft

Domain-specific elementary operators

1. *Actuate fuel/oxygen test switch
2. *Observe fuel quantity indicator
3. *Locate ENGINE IGNITION circuit breaker
4. Decide: If *ENGINE IGNITION circuit breaker not engaged
5. *Locate R AUTOSYN INST circuit breaker
6. Decide: If not *R AUTOSYN INST circuit breaker not engaged
7. *Locate IGNITION INVERTER circuit breaker
8. Decide: If *IGNITION INVERTER circuit breaker not engaged
9. Decide: If *engine started
10. *Locate fuel flow indicator
11. *Advance throttle past IDLE
12. Decide: If *fuel flow increases or *engine lights

Domain-specific complex operators

1. {Look for spark at AB plug}
2. {Listen for sparks from main plug}
3. {Place metal on stick over end of plug}
4. Decide: If {fuel quantity indicator shows correct value}
5. Decide: If {altitude problem},
6. Decide: If {mist not present}
7. Decide: If {fuel flow high enough}
8. Decide: If {pressure} >= 14 psig
9. Decide: If {all fuel is purged from engine}
10. {Disconnect overspeed governor bypass hose from fuel inlet manifold}
11. {Put cap on fuel inlet manifold fitting}
12. {Connect hose to fitting}
13. Decide: If {aircraft operated outside of limitations}

T38 Engine Troubleshooting Procedures - Analyst: David Kieras, U. Michigan

Goals for which there are no methods provided in the procedures

1. air-motor engine
2. apply external electrical power to aircraft
3. attach brass or steel wool to end of stick
4. check boost pump pressure at inlet manifold
5. check engine drain lines for excessively draining component
6. check engine for freedom of rotation
7. check engine starting air inlet duct and crossover duct check valves for freedom of operation and proper installation
8. check for power at Pin N of engine ignition and accessories disconnect plug
9. check fuel mist discharge
10. check fuel system circuit breakers
11. check ignition time-delay relay for 30+/- 3 sec duration
12. check static inverter
13. get cap
14. get graduated container
15. get stick and brass or steel wool
16. measure pressure
17. measure quantity of fuel discharged from hose in 1 minute
18. obtain Handbook limitations
19. obtain operating information
20. remove engine from aircraft
21. start engine
22. stop air-motoring
23. turn on boost pumps

T38 Engine Troubleshooting Procedures - Analyst: David Kieras, U. Michigan

Method Listing

```
    [6-5] Method for Goal: diagnose engine no start symptom
      1.  [6-5.1] Accomplish Goal: check for bad ignition
      2.  Decide: If Recall that problem is bad ignition, and Return with goal
          accomplished
      3.  [6-5.2] Accomplish Goal: check for fuel flow problem
      4.  Decide: If Recall that problem is low fuel flow, and Return with
          goal accomplished
      5.  [6-5.3] Accomplish Goal: check for starting system defective
      6.  Decide: If Recall that problem is bad starting system, and Return
          with goal accomplished
      7.  [6-5.4] Accomplish Goal: check for altitude limitation problem
      8.  Decide: If Recall that problem is altitude limitation, and Return
          with goal accomplished
      9.  Retain that problem was not found, and Return with goal accomplished

    [6-5.1] Method for Goal: check for bad ignition
          returns that problem is/is not bad ignition
      1.  [1] Accomplish Goal: check igniter plugs
            Method provided in line
      2.  Decide: If Recall that plugs are firing,
            Retain that problem is not bad ignition and Return with goal
            accomplished
      3.  [2] Accomplish Goal: check static inverter electrical power
            flow of control not very clear here
            Method provided in-line
      4.  [3] Decide: If Recall that inverter is not producing electrical
          power then
            Accomplish Goal: check start circuit breakers
      5.  [3] Decide: If Recall that starting circuit breakers are ok, and
          Recall that static inverter is not producing electrical power then
            Accomplish Goal: check static inverter
      6.  Decide: If Recall that static inverter is bad, then
            Retain that problem is bad ignition and Return with goal
            accomplished
      7.  [4--6] Decide: If Recall that starting circuit breakers are ok, and
            rationale or flow of control not at all clear here - how does
            applying external power change the situation?
            Recall that static inverter is producing electrical power then
            Accomplish Goal: check ignition with external power
      8.  Decide: If Recall that problem in aircraft electrical system, then
            Retain that problem is bad ignition
      9.  Decide If Recall that problem in engine, then
            Retain that problem is bad ignition,
            Accomplish Goal: remove engine from aircraft,
            and Return with goal accomplished
      10. Retain that problem is not bad ignition, and Return with goal
          accomplished

    [6-5.1.1] Method for Goal: check igniter plugs
      1.  Accomplish Goal: check AB plug
            "AB" apparently means afterburner
      2.  Accomplish Goal: check main plug
      3.  Decide: If Recall that main plug is firing and Recall that AB plug
          is firing,
            Retain that plugs are firing and Return with goal accomplished
    [6-5.1.1] Method for Goal: check AB plug
      1.  ?Watch up tailpipe
      2.  ?Tell assistant try to start engine
      3.  ?{Look for spark at AB plug}
      4.  Count sparks
      5.  Decide if 3 per 2 sec, Retain that AB plug is firing
            Else Retain that AB plug is not firing
      6.  Return with goal accomplished

    [6-5.1.1] Method for Goal: check main plug
      1.  ?Accomplish Goal: obtain shorting stick
      2.  ?Accomplish Goal: apply shorting stick to AB plug
      3.  ?Listen at tailpipe
      4.  ?Tell assistant to try to start engine
      5.  {Listen for sparks from main plug}
      6.  Count sparks
      7.  Decide: If count 3 per 2 sec, Retain that main plug is firing
      8.  Return with goal accomplished

    [6-5.1.1] ?Method for Goal: obtain shorting stick
      1.  Decide: If shorting stick already available, get & Return with goal
          accomplished
      2.  Accomplish Goal: get stick and brass or steel wool
      3.  Accomplish Goal: attach brass or steel wool to end of stick
      4.  Return with goal accomplisheD

    [6-5.1.1] ?Method for Goal: apply shorting stick to AB plug
      1.  Stand at tailpipe
      2.  Hold stick in hand
      3.  Reach into engine
      4.  {Place metal on stick over end of plug}
      5.  Return with goal accomplished

    [6-5.1.2] Method for Goal: check static inverter electrical power

      1.  *Actuate fuel/oxygen test switch
      2.  *Observe fuel quantity indicator
      3.  Decide: If {fuel quantity indicator shows correct value}, then
            Retain that static inverter is producing electrical power and
               Return with goal accomplished
            Else, Retain that static inverter is not producing electrical
               power and Return with goal accomplished

    [6-5.1.3] ?Method for Goal: check starting circuit breakers
      1.  *Locate ENGINE IGNITION circuit breaker
      2.  Decide: If *ENGINE IGNITION circuit breaker not engaged, then Retain
          that ENGINE IGNITION circuit breaker was not engaged
               Else Retain that ENGINE IGNITION circuit breaker was engaged
      3.  *Locate R AUTOSYN INST circuit breaker
      4.  Decide: If not *R AUTOSYN INST circuit breaker not engaged, then
          Retain that R AUTOSYN INST circuit breaker was not engaged
               Else Retain that R AUTOSYN INST circuit breaker was engaged
      5.  *Locate IGNITION INVERTER circuit breaker
      6.  Decide: If *IGNITION INVERTER circuit breaker not engaged, then
          Retain that IGNITION INVERTER circuit breaker was not engaged
            Else Retain that IGNITION INVERTER circuit breaker was engaged
      7.  Decide: If no Recall that a circuit breaker was not engaged, then
            Retain that starting circuit breakers are ok
      8.  Return with goal accomplished
```

```
[6-5.1.4] Method for Goal: check ignition with external power
  1.  [4] Accomplish Goal: apply external electrical power to aircraft
  2.  [4] Accomplish Goal: check igniter plugs
  3.  [5] Accomplish Goal: check for power at Pin N of engine ignition and
      accessories disconnect plug
  4.  [6] Decide: If Recall that power not at Pin N, then
        Accomplish Goal: check aircraft electrical system
  5.  [7] Decide: If Recall that power present at Pin N, then
        Retain that problem is in engine
  6.  Return with goal accomplished

[6-5.2] Method for Goal: check for fuel flow problem
      returns that problem is/is not low fuel flow
      QUESTION: What is the correct flow of control here? First step seems
      to be a symptom-obtaining step, while step [3] can fix the problem.
      Flow of control as shown here is just a guess - could easily be
      incorrect!
  1.  [1] Accomplish Goal: determine if fuel flow is low
  2.  Decide: If Recall that fuel flow is not low, then Retain that
      problem is not low fuel flow and Return with goal accomplished
  3.  [2] Accomplish Goal: check throttle rigging
  4.  Decide: If Recall that throttle rigging was bad, Retain that problem
      is low fuel flow and Return with goal accomplished
  5.  [3] Accomplish Goal: try start with boost pumps
  6.  Decide: If *engine started, then
        Retain that problem was air in fuel system
        Retain that problem is low fuel flow and
        Retain that problem is now corrected
        Return with goal accomplished
  7.  Decide: If Recall that problem is now corrected, and Return with
      goal accomplished
  8.  [4] Accomplish Goal: check fuel system circuit breakers
  9.  Decide: If Recall that breakers were not ok, Retain that problem is
      low fuel flow and Return with goal accomplished
  10. [5] Decide: If {altitude problem},
        Accomplish Goal: check boost pump pressure at inlet manifold
  11. Decide: if boost pump pressure is bad, Retain that problem is low
      fuel flow, and Return with goal accomplished
  12. [6] Accomplish Goal: check fuel shutoff valve is open
  13. Decide: If Recall that valve is closed,
        Retain that problem is low fuel flow and Return with goal
        accomplished
  14. [7] Accomplish Goal: check engine drain lines for excessively
      draining component
  15. Decide: If Recall that component draining excessively, then Retain
      that problem is low fuel flow and Return with goal accomplished
  16. [8] Accomplish Goal: check overspeed governor for internal leakage
        Method provided inline
  17. Decide: If Recall that overspeed governor is bad, then Retain that
      problem is low fuel flow and Return with goal accomplished
  18. [9] Retain that engine is bad,
        Retain that problem is low fuel flow,
        Accomplish Goal: remove engine from aircraft,
        and Return with goal accomplished
```

```
[6-5.2.1] Method for Goal: determine if fuel flow is low
  1. Accomplish Goal: air-motor engine
       Method in section II
  2. Accomplish Goal: check fuel mist discharge
  3. Decide: If {mist not present} Retain that fuel flow is low
  4. Accomplish Goal: check fuel flow indicator
  5. Return with goal accomplished

[6-5.2.1] Method for Goal: check fuel flow indicator
  1. *Locate fuel flow indicator
  2. Decide: If {fuel flow high enough} then Retain that fuel flow is ok
       Else Retain that fuel flow is low
  3. Return with goal accomplished

[6-5.2.2] Method for Goal: check throttle rigging
  1. *Advance throttle past IDLE
  2. Decide: If *fuel flow increases or *engine lights , then
       Accomplish Goal: check throttle rigging
  3. Return with goal accomplished

[6-5.2.3] Method for Goal: try start with boost pumps
  1. Accomplish Goal: apply external electrical power to aircraft
  2. Accomplish Goal: turn on boost pumps
  3. Accomplish Goal: start engine

[6-5.2.5] ?Method for Goal: check boost pump pressure at inlet manifold
  1. *turn on boost pump
  2. *open throttle
  3. Accomplish Goal: measure pressure
  4. Decide: If {pressure} >= 14 psig, Retain that boost pump pressure is
     ok
       else Retain that boost pump pressure is bad
  5. Return with goal accomplished

[6-5.2.6] Method for Goal: check fuel shutoff valve is open
  1. *Locate fuel shutoff valve
  2. *Retain state of valve
  3. Return with goal accomplished

[6-5.2.8] Method for Goal: check overspeed governor for internal leakage
  1. [a] Accomplish Goal: set up overspeed governor bypass hose for check
  2. [b] Accomplish Goal: check flow rate from bypass hose
  3. [c] *Move throttle to off
  4. Decide: If {all fuel is purged from engine}, then
       QUESTION how do you tell that all fuel is purged from engine?
       Accomplish Goal: stop air-motoring
       Else wait
  5. [d] Accomplish Goal: restore bypass hose
  6. Return with goal accomplished

[6-5.2.8.a] Method for Goal: set up overspeed governor bypass hose for
check
  1. Accomplish Goal: get graduated container
  2. Accomplish Goal: get cap
  3. {Disconnect overspeed governor bypass hose from fuel inlet manifold}
  4. Run hose into a graduated container
  5. {Put cap on fuel inlet manifold fitting}
  6. Return with goal accomplished
```

```
[6-5.2.8.b] Method for Goal: check flow rate from bypass hose
  1.  Accomplish Goal: air-motor engine at 14% rpm
        QUESTION: 14% is given as normal maximum rate - why made explicit
        here?
  2.  *advance throttle to IDLE
  3.  Accomplish Goal: measure quantity of fuel discharged from hose in
      minute
  4.  Decide: If rate > 26 fl oz min, 80 phr, then Retain that rate is too
      high and Retain that overspeed governor is bad
        Else overspeed governor is ok
  5.  Return with goal accomplished
```

[6-5.2.8.d] Method for Goal: restore bypass hose

```
  1.  *Remove cap from fitting
  2.  {Connect hose to fitting}
  3.  Return with goal accomplished
```

[6-5.3] Method for Goal: check for starting system defective

```
      returns that problem is/is not bad starting system
  1.  [1] Accomplish Goal: check jet engine starter
  2.  Decide: If Recall that jet engine starter is bad, Retain that
      problem is bad starting system, and Return with goal accomplished
  3.  [2] Accomplish Goal: check ignition time-delay relay for 30+/- 3 sec
      duration
  4.  Decide: If Recall that ignition time-delay relay is bad, Retain that
      problem is bad starting system, and Return with goal accomplished
  5.  [3] Accomplish Goal: check diverter value for proper positioning
      during start cycle
  6.  Decide: If Recall that diverter value is bad, Retain that problem is
      bad starting system, and Return with goal accomplished
  7.  [4] Accomplish Goal: check air duct from jet engine starter for
      blockage or kinking
        Method not provided, but duct is a visible tube and so it should
        not be domain-specific to tell if it is blocked or kinked.
  8.  Decide: If Recall that air duct is bad, Retain that problem is bad
      starting system, and Return with goal accomplished
  9.  [5] Accomplish Goal: check engine starting air inlet duct and
      crossover duct check valves for freedom of operation and proper
      installation
        How does this differ from step 3 above?
  10. Decide: If Recall that engine starting air inlet duct and crossover
      duct check valves is bad, Retain that problem is bad starting
      system, and Return with goal accomplished
  11. [6] Accomplish Goal: check engine for freedom of rotation
  12. Decide: If Recall that engine is bad, Retain that problem is bad
      starting system,
        Accomplish Goal: remove engine from aircraft
  13. Retain that problem is not bad starting system, and Return with goal
      accomplished
```

[6-5.3.1] Method for Goal: check jet engine starter

```
      QUESTION: these data may have already been obtained, but otherwise
      seem to require that the starter be applied to a known good engine?
  1.  Accomplish Goal: start jet engine with starter
  2.  Decide: If produced 14% rpm within 15 sec
        and continued to supply air for
           how do you tell that it is continuing to supply air?
           10 sec after lightoff or at 40% rpm
```

```
          then Retain that starter is good
          else Retain that starter is bad
    3. Return with goal accomplished
```

[6-5.4] Method for Goal: check for altitude limitation problem

returns that problem is/is not altitude limitation

```
    1. Accomplish Goal: obtain operating information
    2. Accomplish Goal: obtain Handbook limitations
    3. Decide: If {aircraft operated outside of limitations}, Retain that
       problem is altitude limitation, and Return with goal accomplished
    4. Retain that problem is not an altitude limitation, and Return with
       goal accomplished
```